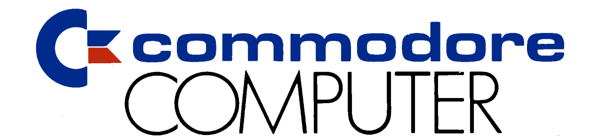
SIMON'S BASIC



Modul-Version
über 100 zusätzliche Basic Befehle
HANDBUCH



S I M O N S - B A S I C

BASIC - Erweiterung für Commodore 64

BEDIENUNGSANLEITUNG

für die

Modul-Version

Commodore Büromaschinen GmbH
Postfach 710126
D-6000 Frankfurt/M 71

INHALTSVERZEICHNIS

ABSCHNITT EINS

Einführung in die Bedienung von SIMON's-BASIC

SIMON's-BASIC ist eine BASIC-Erweiterung für den Commodore 64. Diese Erweiterung wird mit

```
LOAD "*",8  (RETURN)
```

geladen und mit

```
RUN  (RETURN)
```

gestartet.

Das System meldet sich mit

```
*** EXPANDED CBM V2 BASIC ***
30719 BASIC BYTES FREE
```

Dem Benutzer stehen nun über 100 neue BASIC-Befehle zur Verfügung. Jeder Befehl wird in dieser Anleitung erklärt und die Benutzung an vielen Beispielen gezeigt.

Modulversion

Wenn Sie mit SIMON'S BASIC arbeiten wollen, gehen Sie bitte folgendermaßen vor:

1. Schalten Sie Ihren COMMODORE 64 aus.
2. Stecken Sie das Modul „SIMON'S-BASIC" in den Modulsteckplatz Ihres COMMODORE 64 (Beschriftung des Moduls nach oben).
3. Schalten Sie Ihren COMMODORE 64 ein, das System meldet sich mit:
   *** EXTENDED CBM V2 BASIC ***
   30719 BASIC BYTES FREE

Jetzt stehen Ihnen über 100 neue BASIC-Befehle zur Verfügung.

Wichtig:

Stecken bzw. entfernen Sie niemals das Modul, wenn Ihr COMMODORE 64 eingeschaltet ist!!!

ABSCHNITT ZWEI

Programmier-Hilfen

2.1 Einführung

SIMON's-BASIC bietet verschiedene Befehle, die beim Eingeben und Anfügen von BASIC-Programmen sehr hilfreich sind. Dabei ist es unbedeutend, ob die Programme Befehle des SIMON's-BASIC verwenden oder nicht. Der Befehl "KEY" ermöglicht es, die Funktionstasten des Commodore 64 zu programmieren. "DISPLAY" listet die Kommandos der Funktionstasten. "AUTO" und "RENUMBER" übernimmt die automatische Zeilennummerierung. "MERGE" verbindet ein gespeichertes BASIC-Programm mit einem Programm im Speicher des C=64.

2.2 Befehle für die Benutzung der Funktionstasten

2.2.1. KEY

Format : KEY Nummer, "Befehl"
Ziel : Die Funktionstaste wird mit einem Befehl belegt.

Der Befehl KEY ermöglicht es, die Funktionstasten mit den gewünschten Befehlen zu belegen und diese Belegung auch zu ändern. Die Nummer im Format des Befehls ist identisch mit der Nummer der Funktionstaste (1 - 16). Der Befehl muß in Anführungszeichen gesetzt werden und darf nicht mehr als 15 Zeichen umfassen.

Erreichen der Funktionstasten:

F1, F3, F5, F7	: drücken	F1, F3, F5, F7
F2, F4, F6, F8	: drücken SHIFT	F1, F3, F5, F5
F9, F11,F13,F15	: drücken C= Taste	F1, F3, F5, F7
F10,F12,F14,F16	: drücken SHIFT C= Taste	F1, F3, F5, F7

Beispiel
Eingabe : KEY 8, "MOB SET" (RETURN)
Ergebnis : Der Befehl MOB SET wird nach drücken der F8-Taste auf dem Bildschirm angezeigt.
Löschen der KEY's: KEY 8, " " (RETURN)

2.2.2. Anhängen von (RETURN)

Bei bestimmten Befehlen ist es sinnvoll, sie gleich mit °RETURNµ abzuschließen, damit dieser Befehl sofort ausgeführt wird. Dazu geht man wie folgt vor:

a) gewünschte Belegung einer Funktionstaste mit dem Befehl KEY eingeben, nicht mit (RETURN) abschließen.

b) + CHR$(13) anhängen und (RETURN)
Der gewünschte Befehl wird nach Drücken der Funktionstaste angezeigt und gleich ausgeführt.
Beispiel: KEY 7, "LIST" + CHR$(13) (RETURN)
Ergebnis: Nach drücken der Funktionstaste F7 erscheint LIST auf dem Bildschirm und wird ausgeführt, d.h. das Programm wird automatisch aufgelistet.

2.2.3 DISPLAY

FORMAT: DISPLAY
ZIEL: Belegung der Funktionstasten anzeigen.
BEISPIEL: Eingabe

```
DISPLAY (RETURN)
KEY1, ""
KEY2, ""
KEY3, ""
usw.bis KEY16,""
```

2.3 AUTO

FORMAT: AUTO Startnummer, Schrittweite
ZIEL: Automatische Zeilennummerierung mit vorgewählter Schrittweite.

Nach der Eingabe des Befehls AUTO erscheint die ersten Zeilennummer auf dem Bildschirm. Nun kann die gewünschte Basic-Zeile eingegeben werden; sie wird mit (RETURN) abgeschlossen. Es ererscheint eine neue, um die Schrittweite erhöhte Zeilennummer. Um diese Funktion zu beenden, drückt man nach einer neuen Zeilennummer (RETURN).

Beispiel

```
Eingabe:  AUTO 10,5 (RETURN)
Anzeige:  10
Eingabe:  GET A$ (RETURN)
Anzeige:  10 GET A$
          15
Eingabe:  IF A$= "" THEN 10 (RETURN)
Anzeige:  10 GET A$
          15 IF A$= "" THEN 10
          20
Eingabe:  (RETURN)   (AUTO - MODE verlassen)
```

2.4 RENUMBER

FORMAT: RENUMBER erste Zeilennummer, Schrittweite
ZIEL: Umnumerierung der Zeilennummern bei Programmen.

RENUMBER übernimmt automatisch die Umnumerierung der Zeilennummern. Das Programm beginnt nach Ausführung des Befehls mit der ersten Zeilennummer, die eingegeben wurde. Die folgenden Zeilennummern sind jeweils um die eingegebene Schrittweite erhöht.

Dieser Befehl ist besonders sinnvoll, um in einem Programm Platz für weitere BASIC-Zeilen zu schaffen.

ACHTUNG: RENUMBER verändert keine Sprungadressen der Befehle GOTO und GOSUB
SIMON's BASIC bietet jedoch auch hierfür eine Möglichkeit (s. Abschnitt 9).

BEISPIEL: Neue Zeilennumerierung des folgenden Programms

```
1 PRINT"▯"
2 FOR X=0 TO 15
3 : POKE 53280,X
4 : PAUSE 1
5 NEXT X
```

Eingabe:
RENUMBER 100, 10 (RETURN)
LIST (RETURN)

Anzeige:

```
100 PRINT"␣"
110 FOR X=0 TO 15
120 : POKE 53280,X
130 : PAUSE 1
140 NEXT X
```

2.5 PAUSE

FORMAT: PAUSE "Kommentar", Anzahl von Sekunden
oder: PAUSE Anzahl von Sekunden
Ziel: Verzögern der Programmausführung
PAUSE bietet die Möglichkeit, die Programmausführung an einer bestimmten Stelle für eine bestimmte Dauer zu verzögern. Diese PAUSE kann durch drücken von (RETURN) abgebrochen werden. Zusätzlich besteht die Möglichkeit, einen Kommentar auf den Bildschirm zu bringen.
Beispiel: Programmverzögerung um 10 Sekunden
Eingabe:

```
105 PAUSE 10
135 PAUSE "DRÜCKEN SIE RETURN, UM DAS PROGRAMM FORTZUSETZEN", 10
```

2.6 LIN

FORMAT: LIN
ZIEL: Bestimmen der Bildschirmzeile, in der sich der Cursor befindet.
BEISPIEL:

```
90 PRINT"␣"
100 FOR I=1 TO 5
110 : A=LIN
120 : PRINTLIN,A
130 NEXT I
```

2.7 CGOTO

FORMAT: CGOTO Ausdruck
oder: CGOTO Operand Operator Variable
ZIEL: Sprung im Programm zu einer veränderbaren Zeilennummer
Der CGOTO-Befehl ermöglicht es, in eine Zeile zu springen, die erst bei der Ausführung des Befehls berechnet wird.
Beispiel:

```
100 FOR I=1 TO 5
110 : CGOTO I*10 + 130
130 END
140 PRINT"I=1" : NEXT I
150 PRINT"I=2" : NEXT I
160 PRINT"I=3" : NEXT I
170 PRINT"I=4" : NEXT I
180 PRINT"I=5" : NEXT I
```

2.8 RESET

FORMAT: RESET Zeilennummer
ZIEL: Setzen des DATA-Zeigers auf den ersten Wert einer bestimmten DATA-Zeile.
Im Standard-BASIC werden alle DATA-Anweisungen nacheinander gelesen. Es besteht nur die Möglichkeit, durch den Befehl RESTORE den DATA-Zeiger auf den ersten Wert der ersten DATA-Anweisung zu setzen.
RESET bietet die Möglichkeit, den DATA-Zeiger auf den ersten Wert einer bestimmten DATA-Zeile zu setzen. Beim nächsten READ-Befehl wird dieser Wert gelesen und der DATA-Zeiger um eine Stelle nach hinten versetzt.

Beispiel:

```
100 REM ******************************
110 REM *** BEISPIEL :             ***
120 REM *** RESET                  ***
130 REM ******************************
140 PRINT"␣"
150 PRINT"BITTE WAEHLEN :"
160 PRINT"1.) ZAHL 2.)BUCHSTABE 3.) FARBE
170 INPUT Z
180 IF Z<1 OR Z>3 THEN 160
190 IF Z=1 THEN RESET 300
200 IF Z=2 THEN RESET 310
210 IF Z=3 THEN RESET 320
220 FOR I=1 TO 4
230 : READ A$:PRINTA$
240 NEXT I
250 GOTO 150
300 DATA 3,5,7,9
310 DATA A,S,D,F
320 DATA ROT,BLAU,GELB,SCHWARZ
```

2.9 MERGE

FORMAT: MERGE "Programmname", Gerätenummer
ZIEL: Laden eines vorher gespeicherten Programms in den Arbeitsspeicher des Commodore 64.
Ein im Arbeitsspeicher vorhandenes Programm bleibt erhalten. Dabei werden Programmname und Gerätenummern entsprechend dem LOAD-Befehl verwendet.

Achtung: Bei Verwendung des MERGE-Befehls dürfen in dem Programm, das geladen wird, und dem Programm im Arbeitsspeicher keine gleichen Zeilennummern vorkommen. Es empfiehlt sich, das Programm im Arbeitsspeicher mit dem Befehl RENUMBER (2.4) umzunumerieren.

2.10 Hilfen zum Listen der Programme

2.10.1 PAGE

FORMAT: PAGE n
ZIEL: Teilen eines Programms in Abschnitte zu n+1 Zeilen

PAGE bietet die Möglichkeit, die Anzahl der Bildschirm-Zeilen vorzuwählen. Nachdem dieser Befehl ausgeführt wurde, zeigt der Befehl LIST genau n+1 Bildschirmzeilen des Programms an. Jeder weiterer Abschnitt wird nach drücken der (RETURN)-Taste gelistet. Bei n=0 hat der PAGE-Befehl keine Wirkung auf den LIST-Befehl. Das LISTEN kann mit der RUN/STOP-Taste unterbrochen werden.

Achtung: Der Parameter n bezieht sich auf Bildschirmzeilen, nicht auf Programmzeilen.

2.10.2 OPTION

FORMAT: OPTION n
ZIEL: Hervorheben aller neuen Befehle des SIMON's-BASIC

OPTION mit dem Parameter n=10 bewirkt eine reverse Darstellung der SIMON's-BASIC-Befehle, wenn das Programm gelistet wird. Ein beliebiger Parameterwert schaltet das Hervorheben wieder ab.

2.10.3 DELAY

FORMAT: DELAY n
ZIEL: Verändern der Geschwindigkeit, mit der ein Programm gelistet wird.

Während der Ausführung von LIST besteht die Möglichkeit, die Geschwindigkeit, mit der die Zeichen auf den Bildschirm kommen, zu verändern. Hierzu gibt man vor dem LIST-Befehl den Befehl DELAY n (RETURN) ein. Der Parameter n kann Werte von 1 bis 255 annehmen; n=1 höchste Geschwindigkeit, n=255 niedrigste Geschwindigkeit.
Nach LIST (RETURN) wird das Programm mit der gewünschten Geschwindigkeit gelistet, wenn die SHIFT -Taste gedrückt ist.
Weiterhin besteht die Möglichkeit, den LIST-Vorgang durch drücken der CTRL -Taste zu verlangsamen oder ihn durch drücken der C= Taste zu unterbrechen.

2.11 FIND

FORMAT: FIND Basic-Code
oder: FIND Character String
ZIEL: Suchen eines BASIC-Codes oder Character String und Anzeigen der Zeilennummern, in der der Code bzw. String steht.

Der Befehl FIND durchsucht das Programm nach dem angegebenen BASIC-Code oder Character String und zeigt auf dem Bildschirm die Zeilennummern an, in denen das Gesuchte steht.

ACHTUNG: Jede Leerstelle (SPACE) nach FIND wird als Zeichen angesehen, nachdem gesucht werden soll!

2.12 Hilfen zur Fehlerbeseitigung

2.12.1 TRACE

FORMAT: TRACE 10
bzw: TRACE 0
ZIEL: Anzeige der Zeilennummer, die gerade abgearbeitet wird.

Der Befehl TRACE 10 wird eingegeben, bevor das Programm gestartet wird. Wenn das Programm ausgeführt wird, erscheint in der rechten oberen Ecke ein Fenster, in dem die sechs letzten Zeilennummern angezeigt werden. Die Zeilennummern rollen automatisch, so daß die unterste immer der zuletzt ausgeführten BASIC-Zeile entspricht.
Die Zeilennummern werden im HIRES oder MULTI-COLOUR-Modus nicht sichtbar.
Das Fenster überdeckt den normalen Bildschirminhalt.

Der TRACE-Befehl wird durch Eingabe von TRACE 0 beendet.

2.12.2 RETRACE

FORMAT: RETRACE
ZIEL: Nochmalige Anzeige des TRACE-Fensters

Wird im TRACE-Modus ein Programm unterbrochen und der Bildschirm gelöscht, dann kann das Fenster mit den letzten Zeilennummern durch den Befehl RETRACE nochmal dargestellt werden.

2.13 DUMP

FORMAT: DUMP
ZIEL: Listet die Variablen eines Programms und ihre momentanen Werte.

Nach dem Befehl DUMP werden alle Variablen mit ihren momentanen Werten in der Reihenfolge, in der sie im Programm vorkommen, gelistet. Es besteht die Möglichkeit, das Listen durch drücken der CTRL -Taste zu verzögern.

2.14 COLD

FORMAT: COLD
ZIEL: Bringt den COMMODORE 64 in den Ausgangzustand.

Nach COLD (RETURN) wird der Computer in den Ausgangszustand von SIMON's-BASIC gesetzt und meldet sich wie nach dem Laden des SIMON's-BASIC.

ACHTUNG: Das Programm im Arbeitsspeicher geht verloren. Es kann nur gerettet werden, wenn man bevor man eine neue BASIC-Zeile eingibt, den Befehl OLD ausführt.

2.15 Hilfen zum Programmschutz

2.15.1 Einführung

SIMON's-BASIC bietet zwei Befehle, um bestimmte Zeilen des BASIC-Programms zu schützen.
Damit wird verhindert, daß unbefugte Personen Teile des Programms einsetzen können (z.B. KENNWORTE, KENNZEICHEN ...)

ACHTUNG: Es gibt keine Möglichkeit, diese Befehle aufzuheben, außer der Neueingabe der geschützten Zeilen. Es ist deshalb ratsam, eine ungeschützte Kopie für den eigenen Gebrauch zu speichern.

2.15.2 DISAPA

FORMAT: DISAPA:
ZIEL: Anzeigen, daß die Anweisung in der Programmzeile geschützt werden soll.

Der DISAPA-Befehl wird als Befehl in einer Programmzeile verwendet, um anzuzeigen, daß diese geschützt werden soll. Er setzt automatisch vier Doppelpunkte (:) vor den nachfolgenden Befehl.

Beispiel: Zeile 135 und 150 des folgenden Programms sollen geschützt werden.

```
100 REM ********************************
110 REM *** BEISPIEL :                ***
120 REM *** "DISAPA UND SECURE"       ***
130 REM ********************************
135 DISAPA:::: CODE=123
140 INPUT"CODE";A
150 DISAPA:::: IF A<>123 THEN END
160 PRINT"CODE OK"
```

2.15.3 SECURE

FORMAT: SECURE 0
ZIEL: Schützen aller Programmzeilen, in denen der Befehl DISAPA-Befehl vorkommt.

Der SECURE-Befehl bewirkt, daß der Code in einer Programmzeile, der hinter dem DISAPA-Befehl steht, geschützt wird. Die Programmzeile wird normal ausgeführt.

Beispiel: Das Programm von 2.15.2 wird geschützt

```
100 REM ********************************
110 REM *** BEISPIEL :                ***
120 REM *** "DISAPA UND SECURE"       ***
130 REM ********************************
135
140 INPUT"CODE";A
150
160 PRINT"CODE OK"
```

2.16 OLD

FORMAT: OLD
ZIEL: Aufheben des Befehls NEW

OLD ermöglicht es, ein Programm, das mit dem Befehl NEW gelöscht wurde, zurückzuholen und auszuführen.(Es werden die Zeiger START OF BASIC und END OF BASIC auf den ursprünglichen gesetzt).

Achtung: Der Befehl ist nur wirksam, solange noch keine neue BASIC-Zeile eingegeben wurde.

ABSHNITT DREI

ZEICHENKETTEN-OPERATIONEN

3.1 Einführung

SIMON's-BASIC bietet umfassende String-Operationen, formatierte Ausgabe auf dem Bildschirm und INPUT-Befehle.

In Verbindung mit dem Standard String-Operationen des COMMODORE 64 BASIC wird somit eine vielfältige Kontrolle über Zeichenketten ermöglicht. Alle Befehle dieses Abschnitts können im Edit- oder Programmmodus benutzt werden.

3.2 Zeichenketten

3.2.1 INSERT

FORMAT: INSERT ("einzusetzender String", "äußerer String", p)
ZIEL: Einsetzen eines Strings in einen anderen.

INSERT ermöglicht es, einen String in einen anderen einzubauen. Dabei entsteht ein neuer String.
Der Parameter p gibt die Position an, ab welcher der String eingebaut werden soll.
Der einzusetzende String und der äußere String können sowohl als Stringvariable als auch in Anführungszeichen angegeben werden.
Der INSERT-Befehl kann zusammen mit logischen Vergleichsoperatoren benutzt weden.

Achtung: Der Parameter p muß kleiner sein als die Länge des äußeren Strings. Andernfalls folgt die Fehlermeldung ?INSERT TO LARGE IN Zeile.
Der neu entstandene String darf nicht länger als 255 Zeichen sein, da sonst die Fehlermeldung ?STRING TO LARGE IN Zeile folgt.

Beispiel:
Eingabe:
```
100 A$= INSERT (" IN", "COMMODORE FRANKFURT",9)
110 PRINT A$
RUN (RETURN)
```
Anzeige: **COMMODORE IN FRANKFURT**

Beispiel:
Eingabe ohne NEW

```
120 B$= " IN"
130 C$= "COMMODORE FRANKFURT"
140 D$= INSERT (B$, C$, 9)
150 PRINT D$
RUN (RETURN)
```

Anzeige

```
COMMODORE IN FRANKFURT
COMMODORE IN FRANKFURT
```

Beispiel: Vergleichsoperatoren
Eingabe
NEW

```
100 A=(INSERT(" IN","COMMODORE FRANKFURT",9))="COMMODORE IN FRANKFURT"
110 PRINT A
RUN (RETURN)
```

Anzeige:

```
-1
```

Ergebnis: Ist die Bedingung erfüllt, so ist das Ergebnis -1 (wahr); wenn nicht, so ist es 0 (unwahr).

3.2.2 INST

FORMAT: INST ("überschreibender String", "alter String",p)
ZIEL: Überschreibt einen String ab einer bestimmten Position.

INST überschreibt einen Teil des äußeren Strings mit einem String. Der Parameter p gibt die Position an, ab welcher der äußere String überschrieben wird. Dabei kann der neu entstandene String länger sein als der alte, darf jedoch nicht länger als 255 Zeichen sein.

Beispiel:

```
100 A$= "CBM 4032"
110 B$= INST (" 64 ", A$,4)
120 PRINT A$
130 PRINT B$
RUN (RETURN)
```

Anzeige:

```
CBM 4032
CBM  64
```

3.2.3 PLACE

FORMAT: PLACE ("gesuchter String", "Zeichenkette")
ZIEL: Suchen eines bestimmten Strings.

Place sucht in einer Zeichenkette nach einem bestimmten String. Als Ergebnis wird eine INTEGER-Zahl geliefert, die die Position des ersten gesuchten Zeichens beschreibt. Wird der gesuchte String nicht gefunden, so ist das Ergebnis 0.

Beispiel:

```
100 A$= "1234567890"
110 PRINT PLACE ("6",A$)
```

Anzeige:

```
6
```

3.2.4 DUP

FORMAT: DUP ("Zeichenkette",n)
oder: DUP (Stringvariable,n)
ZIEL: Vervielfachen einer Zeichenkette

Der DUP-Befehl vervielfacht eine Zeichenkette n-mal. Dabei entsteht ein neuer String, der aus n Zeichenketten besteht.

Achtung: n muß so gewählt werden, daß der neue String nicht länger als 255 Zeichen ist (sonst Programmabsturz).

Beispiel:

```
100 A$= DUP ("*",10)
110 B$= "* CBM 64 *"
120 PRINT A$:PRINT B$:PRINT A$
RUN  (RETURN)
```

Ergebnis:

```
**********
* CBM 64 *
**********
```

3.2.5 CENTRE

FORMAT: CENTRE "Zeichenkette"
oder: **CENTRE Stringvariable**
Ziel: Zentrieren einer Zeichenkette in einer Blidschirmzeile
CENTRE bringt eine Zeichenkette genau in die Mitte einer Bildschirmzeile. Die Zeichenkette darf dabei **nicht länger als 39 Zeichen sein.**
Soll nach dem CENTRE-Befehl ein weiterer CENTRE-Befehl **folgen, so muß dazwischen ein PRINT-Befehl erfolgen.**

Beispiel:

```
100 A$= DUP ("*",10)
110 B$= "* CBM 64 *"
120 CENTRE A$:PRINT
130 CENTRE B$:PRINT
140 CENTRE A$
```

Anzeige:

```
**********
* CBM 64 *
**********
```

3.2.6 USE

FORMAT: USE "###.###", "Zeichenkette" : PRINT
oder USE "## text . ### text", "Zeichenkette" : PRINT
oder USE Stringvariable, Stringvariable : PRINT
ZIEL: Formatierte Ausgabe von numerischen Werten.

USE ermöglicht es, einen numerischen Wert in einem vorgegebenen Format auszugeben.
Jedes "Nummer"-Zeichen (#) gibt dabei eine Stelle vor bzw. nach dem Komma (Punkt) an. Es ist auch möglich, Text in eine formatierte Zeile einzubauen. Diese Formatangabe kann auch in Form einer Stringvariablen dargestellt werden.
Die Zahl, die ausgegeben werden soll, muß als Stringvariable oder Zeichenkette dargestellt werden. D.h. Zahlen müssen vor der USE-Ausgabe mit dem Befehl STR$ in einen String umgewandelt werden.
Beispiel:

```
100 REM *******************************
110 REM *** BEISPIEL :             ***
120 REM *** USE                    ***
130 REM *******************************
135 A$=STR$(π * 100)
140 PRINT"π * 100 ="+A$
150 B$="##"
160 FOR X=1 TO 3
170 : C$=DUP(B$,X)
180 : D$=C$ + "." + C$
185 PRINT
190 : PRINT"FORMAT  : ";D$
200 : USE "π * 100 = " + D$,A$ :PRINT
210 NEXT
220 END
```

RUN (RETURN)

```
π * 100 = 314.159265

FORMAT  : ##.##
π * 100 = 14.15

FORMAT  : ####.####
π * 100 =  314.1592

FORMAT  : ######.######
π * 100 =    314.159265
```

3.2.7 AT

FORMAT: PRINT AT (s,z) "Zeichenkette"
oder PRINT "1.String"AT(s,Z)"2.String"
ZIEL: Positionierte Ausgabe eines Strings

AT posititioniert die Ausgabe eines Strings auf eine bestimmte Bildschirmposition. S gibt dabei die Spalte (0-39) und z die Zeile (0-24) an. Es ist möglich, den AT-Befehl mehrmals hintereinander zu gebrauchen.

Beispiel:
```
100 PRINT "(SHIFT) (CLR/HOME)" AT(15,24) "* CBM 64 *" AT(15,0)
"* CBM 64 *"
```

Ergebnis: In der Mitte der obersten und untersten Bildschirmzeile steht der String "* CBM 64 *".

3.3 KONTROLLE DER EINGABE

3.3.1 FETCH

FORMAT: FETCH "Kontrolltaste",l,Variable

FETCH ermöglicht es, bei der Eingabe von der Tastatur nur bestimmte Zeichen zuzulassen und auch die Anzahl der Zeichen zu begrenzen. Die nach dem Anführungszeichen eingegebene Kontrolltaste bestimmt dabei, welches Zeichen eingegeben werden kann; l gibt die Anzahl an, wieviele Zeichen höchstens eingegeben werden sollen. Der eingegebene String oder numerische Wert wird unter einer Variablen abgelegt, deren Namen angegeben weden muß.

Kontrolltaste	zugelassene Zeichen
CLR/HOME:	nur Großbuchstaben (CHR$(65) bis CHR$ (90)
CURSER DOWN:	CHR$ (32) bis CHR$ (64) (Zahlen, Satzzeichen)
CURSER RIGHT:	nur Großbuchstaben und ihre Funktion mit Shift.

Zusätzlich ist es möglich, die Eingabe auf ganz bestimmte Zeichen zu begrenzen, in dem man statt der Kontrolltaste genau die gewünschten Zeichen eingibt.

Beispiel:
```
100 PRINT "FORTSETZUNG (J ODER N) ?
110 FETCH "JN", 1, A$
120 IF A$= "N" THEN PRINT "ENDE" : END
130 PRINT "FORTSETZUNG"
```

Ergebnis: Es ist nur möglich den Buchstaben J oder N einzugeben. Alle anderen haben keine Wirkung.

3.3.2 INKEY

FORMAT: INKEY
ZIEL: Abfrage, welche Funktionstaste gedrück wurde

INKEY ermöglicht es, festzustellen, welche Funktionstaste gedrückt wurde. Das Ergebnis ist die Nummer der Funktionstaste (1 bis 16) oder 0 falls keine Funktionstaste gedrückt wurde.

Beispiel:

```
100 A= INKEY
110 ON A GOSUB 1000,2000
120 GOTO 100
1000 PRINT "F1 WURDE GEDRÜCKT" : RETURN
2000 PRINT "F2 WURDE GEDRÜCKT" : RETURN
```

Ergebnis: Je nach gedrückter Taste wird das entsprechende Ergebnis angezeigt.

3.3.3 ON KEY

FORMAT: ON KEY "Zeichenkette", : GOTO Zeilennummer

oder ON KEY Stringvariable, : GOTO Zeilennummer

ZIEL: Springbefehl zu einer bestimmten Programmzeile

Der Befehl ON KEY veranlaßt den COMMODORE 64 dazu, die Tastatur abzufragen, ob eine der in der Zeichenkette angegebenen Tasten gedrückt wurde. Jeder andere Tastendruck wird ignoriert. Wird einer der in der Zeichenkette angegebenen Buchstaben gedrückt, so wird das Programm in der Zeile fortgeführt, die nach dem GOTO angegeben ist. Die belegte Variable ST speichert den CHR$-Wert der Taste, die gedrückt wurde. (Die vollständige Liste der CHR$-Codes finden Sie in Ihrem COMMODORE 64-Handbuch.)
Der Befehl ist besonders nützlich für Menü-geführte Programme.

ACHTUNG: Wenn der "ON KEY"-Befehl benutzt wird, fragt der COMMODORE 64 die Tastatur ab, bis auf die Taste eines der angebenen Zeichen gedrückt wird. Deshalb ist der Befehl DISABLE anzuwenden, um den "ON KEY"-Befehl rückgängig zu machen.

BEISPIEL: Definieren eines Bereichs von Eingabe-Zeichen.

EINGABE:

```
10 PRINT"<SHIFT CLR/HOME> PRESS A KEY (E TO END)"
20 B$="DGHNVMLPOE"
30 ON KEY B$,:GOTO 50
40 GOTO 20
```

ERGEBNIS: Das Programm wartet, bis auf eine der Tasten gedrückt wird, die in der Zeichenkette angegeben sind.

3.3.4. DISABLE

FORMAT: DISABLE
ZEIL: Aufheben des "ON KEY"-Befehls.

Die Tastaturabfrage, die mit ON KEY gestartet wird, wird mit DISABLE wieder augehoben. Dieser Befehl ist <u>immer</u> dann zu verwenden, wenn der Befehl ON KEY benutzt wurde.
Wird dies einmal vergessen, so geht das Programm auf eine Schleife. d.h. das Programm kehrt jedesmal in die Zeile des "ON KEY"-Befehls zurück, wenn eine der angegebenen Tasten gedrückt wird.

BEISPIEL: Aufheben den "ON-KEY"-Befehls.

EINGABE:
```
10 PRINT "<SHIFT CLR/HOME> PRESS A KEY (E TO END)"
20 B$="DGHNVMLPOE"
30 ON KEY B$,:GOTO 50
40 GOTO 20
50 DISABLE
```

ERGEBNIS: Der "ON KEY"-Befehl wird aufgehoben, wenn eine der in der Zeichenkette angegebenen Tasten gedrückt wird.

3.3.5. RESUME

FORMAT: RESUME

ZIEL: Aktivieren des vorausgehenden "ON KEY"-Befehls.

Mit dem Befehl RESUME wird der zuletzt definierte "ON KEY"-Befehl wieder aufgerufen. Darauf wartet das Programm wieder bis auf die Taste eines der in der Zeichenkette angegebenen Zeichens gedrückt wird.

BEISPIEL: Erweiterung des vorausgehenden Programmes durch "Wiedereinschalten" des Befehls ON KEY:

EINGABE:
```
10  PRINT "<SHIFT CLR/HOME> PRESS A KEY (E TO END)"
20  B$="DGHNVMLPOE"
30  ON KEY B$,:GOTO 50
40  GOTO 30
50  DISABLE
60  A$ = CHR$(ST):X=PLACE(A$,B$)
70  ON X GOTO 80.90,100,110,120,130,140,150,160,170
80  PRINT "IT WAS D": RESUME
90  PRINT "IT WAS G": RESUME
100 PRINT "IT WAS H": RESUME
110 PRINT "IT WAS N": RESUME
120 PRINT "IT WAS V": RESUME
130 PRINT "IT WAS M": RESUME
140 PRINT "IT WAS L": RESUME
150 PRINT "IT WAS S": RESUME
160 PRINT "IT WAS O": RESUME
170 PRINT "IT WAS E": END
```

dann: RUN <RETURN>
V

ANZEIGE: IT WAS V

EINGABE: X

ANZEIGE: keine Anzeige

EINGABE: E

ANZEIGE: IT WAS E
READY

ERGEBNIS: Ein Zeichen, das in dem String definiert wurde, führt zur Ausgabe einer Mitteilung. Jedes andere Zeichen wird ignoriert.

ABSCHNITT VIER

ZAHLENBEHANDLUNG

4.1 Einführung

Dieser Abschnitt beinhaltet verschiedene Befehle, die den Umgang mit Zahlen erleichtern. Dazu gehören Befehle zur INTEGER-Rechnung, Bestimmung der Nachkommastellen, die Umwandlung von Zahlen in andere Zahlensysteme sowie die Ergänzung der BOOLSCHEN-Algebra.

4.2 Zusätzliche arithmetische Operatoren

4.2.1 MOD

FORMAT: MOD (x,y)
ZIEL: Bestimmt den Rest nach einer Integer-Division. Der MOD-Befehl wandelt die Parameter x und y zunächst in zwei Integerzahlen um (ohne Aufrundung). Danach wird eine Integerdivision durchgeführt. Das Ergebnis ist eine Integerzahl, die den Rest der Division darstellt. Der Befehl kann direkt oder im Programm verwendet werden.

Beispiel:
PRINT MOD (11,4) (RETURN)
Anzeige:3
Ergebnis: 11:4=2 Rest 3

4.2.2 DIV

FORMAT: DIV (x,y)
ZIEL: Bestimmung der Vorkommazahl bei einer Integer-Division

Der Befehl DIV wandelt die Parameter x und y in zwei Integerzahlen und führt eine Division aus. Das Ergebnis ist eine Integerzahl, die Vorkommazahl. Der Befehl kann direkt oder im Programm verwendet werden.

Beispiel:
PRINT DIV (11,4) (RETURN)
Anzeige: 2
Ergebnis: 11:4=2 Rest 3

4.2.3 FRAC

FORMAT: FRAC (n)
ZIEL: Bestimmung der Nachkommazahl

Der Befehl FRAC bestimmt die Nachkommazahlen einer Gleitkommazahl. Dabei werden maximal 9 Stellen hinter dem Komma angegeben.

Beispiel:
PRINT FRAC (11/3) (RETURN)
Anzeige: .666666667

4.3 Zahlenumwandlung

4.3.1 % - Binär in Dezimal

Format: PRINT % Binärzahl
ZIEL: Umwandlung einer Binärzahl in eine Dezimalzahl

Der %-Befehl wandelt die angegebene Binärzahl in eine Dezimalzahl. Die Binärzahl muß 8 Ziffern lang sein und darf nur die Ziffern 0 und 1 beinhalten. Ist dies nicht der Fall, dann folgt die Fehlermeldung

? NOT BINARY CHAR

Beispiel:
PRINT % 01011010 (RETURN)
Anzeige: 90

4.3.2 $ - Hexadezimal in Dezimal

FORMAT: PRINT $ Hexadezimalzahl
ZIEL: Umwandlung einer Hexadezimalzahl in eine Dezimalzahl

Der $-Befehl wandelt die angegebene Hexadezimalzahl in eine Dezimalzahl. Die Hexadezimalzahl umfaßt 4 Ziffern, die aus den Werten 0 bis F (0-15) bestehen können.
Werden diese Bedingungen nicht eingehalten, dann folgt die Fehlermeldung

? NOT HEX CHARACTER

Beispiel: PRINT $ABCD (RETURN)
Anzeige: 43981

4.4 EXOR

Format: EXOR (n,n1)
Ziel: Eine exklusiv-Oder-Operation ausführen
Der Befehl EXOR ermöglicht es, eine Exklusiv-Oder-Operation zwischen zwei Zahlen durchzuführen.
Die angegebenen Zahlen werden in Binärzahlen umgewandelt und dann Bit für Bit verknüpft. Das Ergebnis wird in eine Dezimalzahl umgewandelt. Die beiden Parameter n, n1 können Werte zwischen 0 und 65535 annehmen. Es können auch Zahlen verschiedener Zahlensysteme verknüpft werden.

Beispiel:

```
100 REM ******************************
110 REM *** BEISPIEL :             ***
120 REM *** EXOR...                ***
130 REM ******************************
140 FOR X=0 TO 999
150 :  A=PEEK(1024+X)
160 :  IF A=32 THEN 190
170 :  K=EXOR (A,%10000000)
180 :  POKE 1024+X,K
190 NEXT X
200 END
```

Ergebnis: Alle Zeichen auf dem Bildschirm werden invertiert dargestellt.

ABSCHNITT FÜNF

Disketten-Befehle

5.1 Einleitung

SIMON's-BASIC bietet zwei Befehle zur Diskettenbehandlung. Dadurch wird der Zugriff auf Disketten wesentlich vereinfacht.

5.2 DISK

Format : DISK, "Anweisung"
Ziel : Öffnen des Diskettenkanals, Anweisung ausführen, Diskettenkanal schließen.
Der Befehl DISK ersetzt folgende BASIC-Befehle:

OPEN, logische Filenummer, Gerätenummer, Sekundäradresse:
PRINT# logische Filenummer.
Er öffnet einen Kanal zu einer Disketteneinheit, führt den angegebenen Befehl aus und schließt den Kanal wieder.

Beispiel:
Disketten neu formatieren (NEW)
DISK "NO:SIMON's BASIC,01" (RETURN)
Ergebnis: Nach wenigen Minuten ist die Diskette neu formatiert.

5.3 DIR

Format : DIR"$
oder : DIR"$:String*
oder : DIR"$:?String*
Ziel : Listen der Directory oder eines Teils der Directory.

Der DIR-Befehl ersetzt den Befehl LOAD "$",8. Dabei hat man die Möglichkeit, die gesamte Directory zu listen, oder nur bestimmte Filenamen. Jedes Fragezeichen steht für einen beliebigen Buchstaben. Der Stern kürzt den String ab.

Beispiel:
DIR "$ (RETURN)
Ergebnis: Die gesamte Directory wird gelistet.

Beispiel:
DIR "$:??S* (RETURN)
Ergebnis: Es werden alle Filenamen gelistet, deren 3. Buchstabe ein S ist.

ABSCHNITT SECHS

GRAFIK mit SIMON's-BASIC

6.1 Einführung

Dieser Abschnitt beschreibt die Grafik-Befehle des SIMON's-BASIC. Der erste Teil schildert den Aufbau des Bildschirms und erklärt den Unterschied zwischen HIGH-RESOLUTION und MULTI-COLOUR. Dann werden die Farbmöglichkeiten des Commodore 64 gezeigt, und beschrieben, wie die Farbe der Grafik bestimmt wird. Der letzte Teil befaßt sich mit den Grafik-Befehlen und den Möglichkeiten, diese einzusetzen.

6.2 Bildschirmaufbau

Bei der Verwendung der Bildschirmgrafik wird der Bildschirm des Commodore 64 in eine Matrix von maximal 320x200 Punkten aufgeteilt. Jeder Punkt hat eine x- und y-Koordinate. Die linke obere Ecke hat die Koordinaten 0,0. (x,y). Die High-Resolution-Grafik teilt den Bildschirm in in eine Matrix von 320 (x) und 200 (y) Punkten. Die Multi-Colour-Grafik teilt ihn in 160x200 Punkte; der Abstand zweier Punkte in x-Richtung ist also doppelt so groß wie bei High-Resolution-Grafik.

6.3 Commodore 64 Farben

Der Commodore 64 bietet eine Auswahl von 16 verschiedenen Farben. In einer Matrix von 8x8 Punkten können bis zu drei verschiedene Farben gewählt werden. Jeder Farbe ist eine Ziffer zugeordnet:

0	Schwarz	8	Orange
1	Weiß	9	Braun
2	Rot	10	Hellrot
3	Türkis	11	Grau 1
4	Violett	12	Grau 2
5	Grün	13	Hellgrün
6	Blau	14	Hellblau
7	Gelb	15	Grau 3

6.4 Zeichentyp

Alle Zeichen-Befehle des SIMON's-BASIC besitzen ein gemeinsames Merkmal : den Zeichentyp.
Dieser Zeichentyp bestimmt, wie ein Punkt auf dem Bildschirm dargestellt wird.
Die Bedeutung des Zeichentyps.

a) HIGH RESOLUTION MODE

Zeichentyp	Funktion
0	Löscht einen Punkt
1	Zeichnet einen Punkt
2	Invertiert einen Punkt (ist ein Punkt da, so wird er gelöscht und umgekehrt)

b) MULTI-COLOUR MODE

Zeichentyp	Funktion
0	Löscht einen Punkt
1	Zeichnet einen Punkt in Farbe 1(MULTI/LOW COL)
2	dto. 2 "
3	dto. 3 "
4	Invertiert einen Punkt Ein Punkt der Hintergrundfarbe wird in Farbe 3 dargestellt. Ein Punkt der Farbe 1 wird in Farbe 2 dargest. dto. 2 1 dto. 3 wird in Hintergrundfarbe dargestellt.

6.5 GRAFIK-BEFEHLE

6.5.1 HIRES

Format : HIRES zf, hg
Ziel : Bestimmen der Zeichen- und Hintergrundfarbe.

Der HIRES-Befehl schaltet den Bildschirm auf hochauflösende Grafik. Jeder Punkt des Bildschirmes kann nun einzeln gesetzt werden (320x200 Bildschirm-Matrix). Der Parameter zf bestimmt die Farbe, in der ein Punkt gesetzt wird. Der Parameter hg bestimmt die Farbe des Hintergrundes in der ein Punkt gesetzt wird.

Beispiel:

```
 100 HIRES 0,1
1000 GOTO 1000
```

Ergebnis: Die Farbe der Punkte ist schwarz, der Hintergrund weiß.

6.5.2 REC

Format : REC x,y,x1,y1,Zeichentyp
Ziel   : Ein Rechteck zeichnen

Der REC-Befehl zeichet ein Rechteck auf den Bildschirm. Die beiden ersten Parameter x und y bestimmen die Koordinaten der linken oberen Ecke des Rechtecks. Die Parameter x1 und y1 bestimmen die Seitenlängen des Rechtecks.
Der Zeichentyp ist in Abschnitt 6.4. beschrieben.

Beispiel:

```
60 REM ********************************
70 REM *** BEISPIEL :               ***
80 REM *** RECHTECK MIT REC.        ***
90 REM ********************************
100 HIRES 2,1
120 FOR X=5 TO 125 STEP 5
140 : REC X,1.3*X,320-5/2*X,120- 9*X/13,1
160 NEXT X
1000 GOTO 1000
```

Anmerkung: Nach drücken der RUN/STOP-Taste wird der normale Bildschirm angezeigt.

6.5.3 MULTI

Format : HIRES zf,hg : MULTI c1,c2,c3
Ziel   : Den Multi-Colour-Modus einschalten und drei Zeichenfarben bestimmen.

Kommt nach dem HIRES-Befehl der Befehl MULTI, so wird die MULTI-COLOUR-GRAFIK eingeschaltet.
Ein gezeichneter Punkt hat hier die doppelte Breite gegenüber einem Punkt in HIRES-Grafik (160x200 Bildschirm Grafik).
Die Parameter c1 bis c3 legen die drei Farbmöglichkeiten (siehe Farbtabelle in Abschnitt 6.3) fest, die dann durch den Zeichentyp die Farbe eines Punktes bestimmen.

Beispiel :

```
60 REM ********************************
70 REM *** BEISPIEL :               ***
80 REM *** MULIT-COLOUR-RECHTECK    ***
90 REM ********************************
100 HIRES 1,11: MULTI 12, 8,6
120 FOR X=5 TO 60  STEP 1
130 : F=F+1 .
132 : IF F=5 THEN F=0
140 : REC X,1.9*X,160-5/2*X,120- 9*X/13,F
160 NEXT X
1000 GOTO 1000
```

6.5.4 LOW COL

Format : LOW COL c1,c2,c3
Ziel   : Farbwechsel
Der Befehl LOW COL ermöglicht es, 3 weitere Farben für einen Zeichenbefehl bereitzustellen. Die Parameter c1 bis c3 bestimmen diese Farben, entsprechend dem Befehl MULTI.

ACHTUNG : Es müssen immer alle drei Parameter angegeben werden, auch in HIRES-Grafik.

Beispiel :

```
60 REM ********************************
70 REM *** BEISPIEL :               ***
80 REM *** LOW-COLOUR-RECHTECK      ***
90 REM ********************************
100 HIRES 0,2: MULTI 3,4, 9
120 FOR X=5 TO 53  STEP 1
130 : F=F+1
131 : LOW COL F,F+2,F+4
132 : IF F=5  THEN F=0
140 : REC X+ X*1.5,X+2*X,120-2.0*X,130-3.5*X,F
160 NEXT X
1000 GOTO 1000
```

6.5.5 HI COL

Format : HI COL
Ziel : Zurückholen der unter MULTI festgelegten Farben.

Nach dem HI COL-Befehl können die unter MULTI festgelegten Farben wieder verwendet werden (in Verbindung mit dem Zeichentyp).
Beispiel :

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** LOW-COL, HI-COL          ***
130 REM ********************************
140 Z= 5
150 HIRES 0,2: MULTI 10,6,8
160 REPEAT
170 : FOR X=1 TO 3
180 :    REC Z,Z,Z,Z,X
190 :    Z=Z+31
200 :    LOW COL  1,12,0
210 :    REC Z,Z,Z,Z,X
220 :    Z=Z-21
230 :    HI COL
240 : NEXT X
250 : Z=Z-29
260 UNTIL Z>14
270 GOTO 270
```

6.5.6 PLOT

Format : PLOT x,y,Zeichentyp
Ziel : Einen Punkt zeichnen.

Der Befehl PLOT zeichnet einen Punkt auf den Bildschirm. Die Parameter x und y geben die Koordinaten des Punktes an. Der Zeichentyp ist in Abschnitt 6.4 beschrieben.

Beispiel :

```
100 REM ********************************
120 REM *** BEISPIEL :               ***
130 REM *** PLOT...                  ***
140 REM ********************************
150 HIRES 0,1
160 FOR X=0 TO 320 STEP 0.5
170 :   Y=100 + SIN(X/34)*90
180 :   PLOT X,Y,1
185 :   PLOT X,100,1
190 NEXT X
1000 GOTO 1000
```

6.5.7 TEST

Format : variable = TEST (x,y)
Ziel : Die Lage eines Punktes bestimmen.

Der Befehl TEST bestimmt die Lage eines Punktes auf dem Bildschirm. Die Parameter x und y bestimmen die Bildschirm-Koordinaten des TEST-Punktes. Das Ergebnis in der Variablen ist 1, wenn an dieser Stelle ein Punkt sitzt.
Im anderen Fall ist das Ergebnis gleich 0.

Beispiel: Nullstellenbestimmung

```
100 REM ********************************
120 REM *** BEISPIEL :              ***
130 REM *** TEST...                 ***
140 REM ********************************
144 I=1
145 PRINT"■"
150 HIRES 0,1
160 FOR X=0 TO 320 STEP 0.5
170 :   Y=100 - COS(X/33)*90
180 :   PLOT X,Y,1
185 :   T=TEST (X,100)
190 :   IF T=1 THEN GOSUB 1000
200 :   PLOT X,101,1
210 NEXT X
220 PAUSE 2
230 END
1000 REM ******************************
1010 REM *** UNTERPROGRAMM        ***
1020 REM ******************************
1040 N(I)=INT(X)
1050 IF N(I)=N(I-1) THEN RETURN
1060 PRINT"NULLSTELLE BEI X =";N(I)
1070 I=I+1
1080 RETURN
```

6.5.8 LINE

Format : LINE x,y,x1,y1,Zeichentyp
Ziel : Eine Linie zeichnen

Der Befehl-LINE zeichnet eine Linie. x und y sind die Anfangs-Koordinaten, x1 und y1 die End-Koordinaten.
Der Zeichentyp ist in Abschnitt 6.4 beschrieben.

Beispiel :

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** LINE...                 ***
130 REM ********************************
140 HIRES 1,6
150 FOR X=0 TO 320 STEP 8
160 : M =100/320*X
170 : LINE 320-X,M,X,100+M,1
180 NEXT X
200 GOTO 200
```

6.5.9 CIRCLE

Format : CIRCLE x,y,x1,y1,Zeichentyp
Ziel : Zeichnen einer Ellipse

Mit dem Befehl CIRCLE ist es möglich, eine Ellipse zu zeichnen. Die Parameter x und y bestimmen die Mittelpunkt-Koordinaten, x1 den Radius in x-Richtung, y1 den Radius in y-Richtung. Der Zeichentyp ist in Abschnitt 6.4 beschrieben.

ACHTUNG :
Ein Kreis ist eine Sonderform der Ellipse. Damit ein Kreis entsteht, muß eine der Bedingungen erfüllt sein:

HIRES-Grafik	x-Radius = 1,15*y-Radius
MULTI-Grafik	x-Radius = 0,575*y-Radius
Drucker	x-Radius = y-Radius

Beispiel :

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** CIRCLE...               ***
130 REM ********************************
140 HIRES 1,6
150 FOR Y=70 TO 100 STEP 3
160 : X=1.15*Y
170 : CIRCLE 160,    Y,X,Y,1
180 : CIRCLE 160,200-Y,X,Y,1
190 : CIRCLE  45+X,100,X,Y,1
200 : CIRCLE 275-X,100,X,Y,1
210 NEXT Y
220 GOTO 220
```

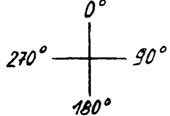
6.5.10 ARC

Format : ARC x,y,sa,ea,i,x1,y1,Zeichentyp
Ziel : Einen Bogen zeichnen
Der ARC-Befehl ermöglicht es einen Bogen zu zeichnen. Die Parameter haben folgende Funktion:

x,y = Mittelpunkt der Ellipse (Kreis) von der ein Bogen gezeichnet werden soll.

sa = Startwinkel

se = Endwinkel



i = Abstand der Winkel, die berechnet werden

Startwinkel + i = nächster Punkt, der berechnet wird. Diese beiden Punkte werden linear verbunden.

x1 = x-Radius der Figur, von der der
y1 = y-Radius Bogen gezeichnet werden soll.

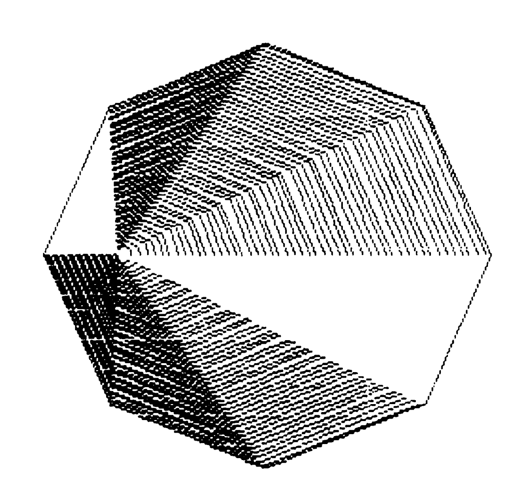
Der Zeichentyp ist in Abschnitt 6.4 beschrieben.

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** ARC ...                  ***
130 REM ********************************
140 HIRES 1,0: MULTI  6,1,1
150 REM ********************************
160 REM *** KREISE ZEICHNEN          ***
170 REM ********************************
180 FOR X= 8 TO 64 STEP 8
190 :  CIRCLE 80,100,X,100,3
200 NEXT X
210 CIRCLE 80,100.72,100,1
220 REM ********************************
230 REM *** ZWEI LINIEN              ***
240 REM ********************************
250 LINE  80,0, 80,200,3
260 LINE 10,100,150,100,1
270 REM ********************************
280 REM *** VIER BOEGEN              ***
290 REM ********************************
300 ARC 80,0,152,207,1,100,30,1
310 ARC 80,200,333,28,1,100,30,1
320 ARC 80,0,142,217,1,100,60,1
330 ARC 80,200,323,38,1,100,60,1
400 GOTO 400
```

Beispiel : Achteck-Pyramide

```
100 REM ******************************
110 REM *** BEISPIEL :              ***
120 REM *** ACHTECK-PYRAMIDE        ***
130 REM ******************************
140 HIRES 1,6
150 FOR X= 5 TO 100  STEP 2
160 : ARC 40+X*.8 ,100,315, 90,45,X*1.2, X,1
170 : ARC 40+X*.8 ,100,135,270,45,X*1.2,X,1
180 NEXT X
190 ARC 120,100,0,360,45,120,100,1
200 GOTO 200
```



6.5.11 ANGL

Format : ANGL x,y,Winkel x1,y1,Zeichentyp
Ziel : Einen Radius einer Figur zeichnen

Der ARC-Befehl ermöglicht es, einen Radius in eine Ellipse einzuzeichnen. Die Ellipse muß dafür nicht auf dem Bildschirm vorhanden sein.
Die Parameter haben folgende Funktion :

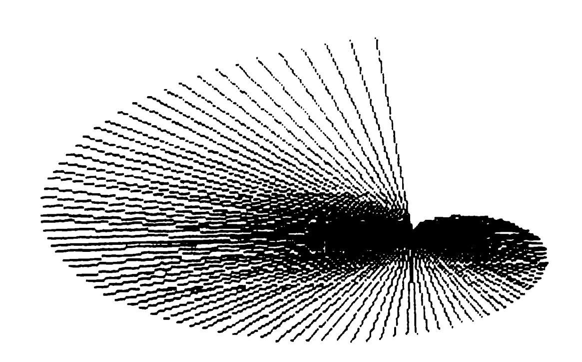
x,y Mittelpunkt der gedachten Ellipse ist Startpunkt des Radius

Winkel = Winkel, unter dem der Radius steht.

x1, y1 Halbachsen der Ellipse von der der Radius gezeichnet werden soll.

Beispiel :

```
100 REM ******************************
110 REM *** BEISPIEL :             ***
120 REM *** ANGL...                ***
130 REM ******************************
140 HIRES 2,1
150 FOR X= 5 TO 358 STEP  3
160 : ANGL 220,125,X,X/1.4,X/3.8,1
170 NEXT X
1000 GOTO 1000
```



6.5.12 PAINT

Format : PAINT x,y,Zeichentyp
Ziel : Eine geschlossene Fläche mit einer Farbe ausfüllen.

Der PAINT-Befehl füllt eine umschlossene Fläche mit einer Farbe aus. Die Farbe wird durch den Zeichentyp bestimmt. Die Parameter x und y geben die Koordinaten eines Punktes innerhalb der zu färbenden Fläche an. Eine Fläche kann mehrmals mit einer Farbe gefüllt werden.

Beispiel :

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** PAINT...                 ***
130 REM ********************************
140 HIRES 0,1: MULTI 5,4,6
150 CIRCLE 80,100,48,78,3
160 ANGL 80,100,120,48,78,1
170 ANGL 80,100,160,48,78,1
180 ANGL 80,100,220,48,78,1
190 ANGL 80,100,330,48,78,1
200 PAINT  90,35,1
210 PAINT  60,60,3
220 PAINT  90,120,2
230 LOW COL 7,4,6
240 PAINT  80,110,1
1000 GOTO 1000
```

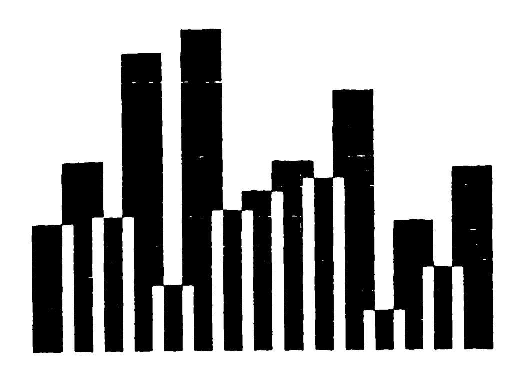
6.5.13 BLOCK

Format : Block x,y,x1,y1,Zeichentyp
Ziel : Ein ausgefülltes Rechteck.

Zusätzlich zu der Möglichkeit, ein Rechteck zu zeichnen und dies mit einer Farbe zu füllen, kann mit dem Befehl BLOCK ein ausgefülltes Rechteck gezeichnet werden. Entscheidend ist dabei die Zeichengeschwindigkeit.
Die Parameter x und y bestimmen die Koordinaten der linken oberen Ecke, x1 begrenzt den Block in x-Richtung, y1 in y-Richtung.
Der Zeichentyp ist in Abschnitt 6.4 beschrieben.

Beispiel :

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** BLOCK...                 ***
130 REM ********************************
140 HIRES 2,2: MULTI 2,1,5
150 X=16
160 REPEAT
170 : FOR F=1 TO 3
180 :    DY=RND(1)*140
190 :    BLOCK X,10+DY ,X+10,160,F
200 :    X=X+8
210 : NEXT F
220 UNTIL X>120
250 GOTO 250
```



6.5.14 DRAW

Format : DRAW "nnnn...9",x,y,Zeichentyp
         DRAW Stringvariable ,x,y,Zeichentyp
Ziel   : Eine Figur zeichnen

Der DRAW-Befehl ermöglicht es, eine beliebige Figur zu entwerfen und diese auf dem Bildschirm zu zeichnen. Die Figur entsteht ähnlich wie das Zeichnen auf dem Papier. Man gibt durch Ziffern (n) an, in welche Richtung sich der "Zeichenstift" bewegt und ob er sichtbar zeichnet oder nicht.
Bedeutung der Ziffern:

0	Bewegung einen Punkt nach	rechts	(nicht sichtbar)
1	dto.	oben	"
2	dto.	unten	"
3	dto.	links	"
4 = 2			
5	dto.	rechts	und einen Punkt setzen
6	dto.	oben	"
7	dto.	unten	"
8	dto.	links	"
9	Ende der Figur		

Die Ziffern werden in Anführungszeichen direkt angegeben oder vorher als String festgelegt. Die Parameter x und y geben die Startkoordinaten an.
Der Zeichentyp ist in Abschnitt 6.4 beschrieben.

Beispiel :

```
100 REM ********************************
110 REM *** BEISPIEL :                ***
120 REM *** DRAW...ROT...             ***
130 REM ********************************
140 HIRES 2,1
150 A$="555555555555555555555555555"
160 D$="888888888888888888888888888"
170 B$="655665666866686886888878878"
180 E$="788778777577575575555565565"
190 C$="378878878878878878878555555"
200 F$="5655655655655655655655659"
205 C1$="5555555555555559"
210 L$=D$+E$+F$
220 R$=A$+B$+C$+C1$
230 FOR U=0 TO 7 STEP 2
240 : ROT U,1
250 : DRAW R$,160,100,1
260 : DRAW L$,160,100,1
270 : DRAW R$,160,100,0
280 : DRAW L$,160,100,0
290 NEXT U
300 GOTO 230
```

6.5.15 ROT

Format : ROT r,s
Ziel : Eine Figur verdrehen und die Größe verändern.

Der Befehl ROT ermöglicht es, eine Figur, die mit dem DRAW-Befehl entworfen wurde, um einen bestimmten Winkel zu drehen. Dieser Winkel wird durch den Parameter r bestimmt :

r	Rotationswinkel
0	0 Grad
1	45 Grad
2	90 Grad
3	135 Grad
4	180 Grad
5	225 Grad
6	270 Grad
7	315 Grad

Er muß im Bereich von 0 bis 7 liegen. Der zweite Parameter bestimmt die Größe der Figur; er kann zwischen 1 und 255 liegen.

ACHTUNG:
Die Größe der Figur hängt auch von der gewählten Grafik ab:

HIRES 320x200 Punkte
MULTI 160x200 Punkte

Beispiel : siehe Abschnitt 6.5.14 (Beispiel DRAW)

6.5.16 CSET

Format : CSET n
Ziel : Auswahl eines Zeichensatzes des Commodore 64
oder zurückholen der letzten Grafik.

Der CSET-Befehl ermöglicht es, den Zeichensatz des Commodore 64 im Programm umzuschalten.

CSET 0 - Großschrift / Grafik-Modus
CSET 1 - Groß- / Kleinschrift-Modus

Weiterhin kann mit diesem Befehl die zuletzt abgebildete Grafik nochmals dargestellt werden.
CSET 2 - HIRES-Grafik
CSET 2 : MULTI c1,c2,c3 - MULTI-Grafik
Die Parameter c1 - c3 bestimmen die neuen Farben.

Beispiel :

```
100 REM ******************************
110 REM *** BEISPIEL :             ***
120 REM *** CSET 0/1               ***
130 REM ******************************
140 PRINT"[CLR]"
150 PRINT AT(12,14)"SIMON'S BASIC"
160 CSET 1:PAUSE 1: CSET 0:PAUSE 1
170 GOTO 140
```

```
100 REM ******************************
110 REM *** BEISPIEL :             ***
120 REM *** CSET 0/2               ***
130 REM ******************************
140 HIRES 2,2: MULTI 2,1,5
150 X=16
160 REPEAT
170 : FOR F=1 TO 3
180 :    DY=RND(1)*120
190 :    BLOCK X,20+DY ,X+10,160,F
200 :    X=X+8
210 : NEXT F
220 UNTIL X>120
230 PAUSE 5: CSET 0
240 PRINT"[CLR][DOWN][DOWN][DOWN][DOWN][DOWN]BITTE EINE TASTE DRUECKEN, DAMIT DIE "
250 PRINT"GRAFIK NOCHMAL ERSCHEINT."
260 PRINT"[DOWN][RVS]E FUER ENDE !"
270 GET ANT$
280 IF ANT$="" THEN 270
290 IF ANT$="E" THEN END
300 CSET 2: MULTI 2,1,5
310 GOTO 230
```

6.5.17 COLOUR

FORMAT: COLOUR br,hg

ZIEL: Bestimmen der Hintergrund- und Bildschirmrahmenfarbe.

Der Befehl COLOUR ermöglicht es, die Hintergrundfarbe (hg) im normalauflösenden Fall und die Rahmenfarbe (br) ebenso bei hochauflösender Grafik zu bestimmen. Die Farben werden durch die entsprechende Ziffer (siehe Abschnitt 6.3) festgelegt. Die Rahmen farbe kann duch einen weiteren COLOUR-Befehl, die Hintergrundfarbe kann bei hochauflösender oder Multi-COLOUR-Grafik durch den HIRES-Befehl geändert werden.

BEISPIEL:

EINGABE: COLOUR 3,6 <RETURN>

ANZEIGE: Ein türkiser Hintergrund mit blauem Rahmen.

6.5.18 NRM

FORMAT: NRM

ZIEL: Schaltet vom Grafik-Bildschirm zum normal auflösenden Bildschirm zurück.

Der Behl NRM löscht eine hochauflösende oder Multi-Colour-Grafik vom Bildschirm und kehrt zum normalen Zustand zurück.

BEISPIEL: Nachdem die Grafik 5 Sekunden angezeigt wurde, kehrt das Programm zum Normalbildschirm zurück.

EINGABE:

```
10 HIRES 0,1: MULTI 0,2,6
30 REC 0,0,40,20,1
40 REC 20,20,40,20,2
50 REC 40,40,40,20,3
60 PAUSE 5
70 NRM

RUN   <RETURN>
```

ANZEIGE: Drei Rechtecke werden für 5 Sekunden angezeigt. Danach erscheint der normale Bildschirm mit READY und einem blinkenden Cursor.

6.6 TEXT IN EINER GRAFIK

6.6.1 CHAR

Format : CHAR x,y,Bildschirmcode, Zeichentyp, Größe
Ziel : Ein einzelnes Zeichen auf einen Grafik-Bildschirm bringen.

Der CHAR-Befehl ermöglicht es, Zeichen auf einem Grafik-Bildschirm darzustellen. Die Parameter haben folgende Funktion:

x,y Koordinaten, die die linke obere Ecke des Zeichens positionieren.

Bildschirmcode - Zeichen, das gePOKED wird.
Zeichentyp - siehe Abschnitt 6.4
Größe - Faktor, um den sich das Zeichen in Vertikalrichtung ausdehnt.

Beispiel :

Ergänzen des Programms CSET 0/2

```
195 :   CHAR x,150,x/8-1,F-1,1
```

6.6.2 TEXT

Format : TEXT x,y,"(CTRLa) String",Zeichentyp,Größe,Abstand
oder : TEXT x,y,"(CTRLb) String",Zeichentyp,Größe,Abstand
Ziel : Eine Zeichenkette auf einen Grafik-Bildschirm bringen.

Der Befehl-Text ermöglicht es, Zeichenketten auf einem Grafik-Bildschirm darzustellen. Die Parameter haben folgende Funktion :

x,y Koordinaten, die die linke obere Ecke des String positionieren.

Der nächste Parameter ist der String selbst oder eine Variable. Das erste Zeichen bestimmt, ob der String im Groß- oder Klein-schrift-Modus dargestellt wird.

(CTRL-a) - Großschrift = Invers A

(CTRL-b) - Kleinschrift = Invers B

Zeichentyp - siehe Abschnitt 6.4
Größe- Faktor, um den sich der String in Vertikalrichtung ausdehnt.
Abstand - bestimmt den Abstand zwischen den einzelnen Zeichen.

Beispiel:
Ergänzen des Programms CSET 0/2:
155TEXT5,5"(CTRLa)BEISPIEL TEXT",1,2,12
156TEXT5,165,"(CTRLb)-OMMODORE 64",3,4,12

(CTRLa)=Invers A
(CTRLb)=Invers B
-       =(SHIFT C)

ABSCHNITT 7

BILDSCHIRMSTEUERUNG

7.1 Einführung

Dieser Abschnitt beschäftigt sich mit der Programmierung des Bildschirms. SIMON's BASIC bietet Befehle zur Farbsteuerung, zum Füllen, Duplizieren sowie Verschieben von Bildschirmbereichen, Bildschirm-Rollen, Speichern und Drucken von Bildschirmdaten.

7.2. FLASH

Format: FLASH f1,speed
Ziel: Blinken einer Bildschirmfarbe

Der Befehl FLASH bewirkt einen dauernden Wechsel einer bestimmten Farbe zwischen INVERS- und NORMALSCHRIFT. Der Parameter f1 bestimmt die Farbe, speed die Geschwindigkeit. Speed kann Werte zwischen 0 und 255 annehmen. Die Einheit ist 1/16 s.
Der Befehl FLASH ist bei HIRES- oder MULTI COLOUR-GRAFIK unwirksam.

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** FLASH                    ***
130 REM ********************************
140 POKE 53280,8 :POKE53281,8
150 PRINT"[CLR]"
160 PRINT" ****************************
170 PRINT" ****************************
180 PRINT" ** BEISPIEL :             **
190 PRINT" ** FLASH                  **
200 PRINT" ****************************
210 PRINT" ****************************
215 FOR F=16  TO 1 STEP-1
220 FLASH 7,F
222 PAUSE 1
225 NEXT F
230 OFF
```

7.3 OFF

Format: OFF

OFF beendet den FLASH-Befehl.

7.4 BFLASH

Format: BFLASH speed,f1,f2
Ziel: Farbwechsel des Bildschirmrahmens

Der Befehl BFLASH bewirkt einen dauernden Farbwechsel des Bildschirmrahmens. Die Parameter f1 und f2 bestimmen die Farben. Der Parameter speed bestimmt die Geschwindigkeit des Farbwechsels (1-255).

7.5 BFLASH 0

Format: BFLASH 0

Der Befehl BFLASH 0 beendet den Befehl BFLASH.

7.6 FCHR

Format: FCHRr,c,w,d,Code
Ziel: Einen Bereich des Bildschirms mit einem Zeichen füllen.

Der Befehl FCHR ermöglicht es, einen bestimmten Bereich des Bildschirms mit einem bestimmten Zeichen zu füllen. Die Parameter r (0-24) und c (0-39) bestimmen die linke obere Ecke des Feldes, w die Anzahl der Spalten, d die Anzahl der Zeilen und Code ist der "POKE-Code" des gewünschten Zeichens.

Beispiel: s. Abscnitt 7.7 (FCHR-FCOL)

7.7 FCOL

Format: FCOL r,c,w,d,Farbe
Ziel: Die Zeichenfarbe in einem bestimmten Bereich bestimmen.

Der Befehl FCOL legt die Zeichenfarbe eines bestimmten Bildschirmbereiches fest. Die Parameter r,c,w,d entsprechen den Parametern des Befehls FCHR (Abschnitt 7.6), der letzte Parameter bestimmt die Farbe.

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** FCHR - FCOL             ***
130 REM ********************************
140 POKE 53280,8:POKE 53281,8
150 PRINT"♥"
160 FCHR 10,10,10,10,10
170 FOR X=10 TO 15 STEP 5
180 : FOR Y=10 TO 15 STEP 5
190 :   FCOL X,Y,5,5,F
200 :   F=F+1
210 : NEXT Y
220 NEXT X
```

7.8 FILL

Format: FILLr,c,w,d,Code,Farbe
Ziel: Einen Bereich des Bildschirms mit einem bestimmten Zeichen in einer bestimmten Farbe füllen.

Der Befehl FILL füllt einen Bereich des Bildschirms mit einem Zeichen in einer vorgegebenen Farbe. Die Parameter r,c,w,d,Code entsprechen den Parametern des Befehls FCHR (Abschnitt 7.6), Farbe bestimmt die Zeichenfarbe.

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** FILL                     ***
130 REM ********************************
140 POKE 53280,1 :POKE 53281,1
150 PRINT"♥"
170 FOR X=0 TO 15
190 :  FILL X,2*X,5,5,X,X
220 NEXT X
230 GOTO 230
```

7.9 MOVE

Format: MOVEr,c,w,d,dr,dc
Ziel: Duplizieren eines Bildschirmbereiches.

Der MOVE-Befehl dupliziert einen Bildschirmbereich. Die Parameter r,c,w,d geben den Bereich an, der dupliziert werden soll, entsprechend Abschnitt 7.6. Die Parameter dr und dc geben die absolute Position an, ab welcher der Bereich dupliziert wird.

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** MOVE                     ***
130 REM ********************************
140 POKE 53280,1 :POKE 53281,1
150 PRINT"♥"
160 FILL 0,0,5,5, 1,10
170 MOVE 0,0,5,5, 0,35
180 MOVE 0,0,5,5,19, 0
190 MOVE 0,0,5,5,19,35
200 GOTO 200
```

7.10 INV

Format: INVr,c,w,d
Ziel: Einen Bildschirmbereich invertieren

Der Befehl INV invertiert alle Zeichen in einem bestimmten Bildschirmbereich. Die Parameter r,c,w,d entsprechen den Parametern im Abschnitt 7.6

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** INV                      ***
130 REM ********************************
140 POKE 53280,8 :POKE53281,8
150 PRINT"[CLR]"
160 PRINT" ****************************
170 PRINT" ****************************
180 PRINT" ** BEISPIEL :             **
190 PRINT" ** INV                    **
200 PRINT" ****************************
210 PRINT" ****************************
215 PAUSE 3
220 INV 2,2,26,4
230 GOTO 230
```

7.11 Bildschirmrollen

Format: Richtung W,sr,sc,er,ec
oder   Richtung B,sr,sc,er,ec
Ziel: Rollen eines bestimmten Bildschirmbereichs

SIMON's-BASIC bietet die Möglichkeit, bestimmte Bildschirmbereiche in vier verschiedene Richtungen zu rollen.
Der erste Parameter bestimmt die Richtung des Bildschirmrollen - LEFT, RIGHT, UP, DOWN. Der zweite Paramter kann entweder W oder B sein:

W Zyklisches Bildschirmrollen
d.h. es geht keine Zeile verloren

B Bildschirmrollen ohne Bildumlauf
d.h. es werden Leerzeilen bzw. -spalten nachgeschoben.

Die Parameter sr,sc,er,ec bestimmen den Bildschirmausschnitt, der gerollt wird:

sr/sc - erste Zeile/Spalte
er/ec - letzte Spalte/Zeile

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :                ***
120 REM *** BILDSCHIRMROLLEN          ***
130 REM ********************************
140 PRINT"␍"
150 FOR X=1 TO 39
160 : Y=INT(10*SIN(X/4))+12
170 PRINTAT(X,Y) "*"
180 NEXT X
190 LEFTW 0,0,20,25 :RIGHTW 0,20,20,25
200 GOTO 190
```

7.12 Abspeichern und Laden von Bildschirmdaten

7.12.1 SCRSV

Format: SCRSV 2,8,2,"Name,s,w"
oder   SCRSV 1,1,1,"Name,s,w"
Ziel: Abspeichern eines LOW-RESOLUTION-Bildschirms

Der Befehl SCRSV ermöglicht es, Bildschirmdaten direkt auf Diskette oder Kassette abzuspeichern.
Die zweite Ziffer hinter dem Befehl gibt die entsprechende Gerätenummer an. Der Name bestimmt den Filenamen, unter dem der Bildschirminhalt abgespeichert wird und unter dem die Daten mit dem Befehl SCRLD (s. Abschnitt 7.12.2) wieder geladen werden können. Mit diesem Befehl können keine HIRES- oder MULTI-COLOUR-Grafiken abgespeichert werden.

Beispiel:

```
100 REM ********************************
110 REM *** BEISPIEL :                ***
120 REM *** SCRSV                     ***
130 REM ********************************
140 PRINT"␍"
150 FILL  6,10,20,4,160,2
160 FILL 10,10,20,4,160,1
170 FILL 14,10,20,4,160,5
180 SCRSV 2,8,2,"SCRSV-DEMO,S,W"
190 GOTO 190
```

7.12.2 SCRLD

Format: SCRLD 2,8,2,"Name"
oder SCRLD 1,1,1,"Name"
Ziel: Laden eines LOW-RESOLUTION-Bildschirms

Mit diesem Befehl können Bildschirmdaten, die mit dem Befehl SCRSV gespeichert wurden, wieder direkt geladen werden. Die Parameter sind entsprechend dem Befehl SCRSV anzugeben.

Beispiel:
Um den Bildschirminhalt des Beispiels SCRSV noch einmal darzustellen:

```
100 REM ********************************
110 REM *** BEISPIEL :               ***
120 REM *** SCRLD                    ***
130 REM ********************************
140 SCRLD 2,8,2,"SCRSV-DEMO"
150 GOTO 150
```

7.13 **Ausdruck von Bildschirmdaten**

7.13.1 COPY

Format: COPY

Der Befehl COPY führt eine Hardcopy einer HI-RES- oder MULTI-COLOUR-Grafik aus. Um auf einem Drucker genau einen Kreis zu bekommen, müssen der x und y-Radius des Befehls CIRCLE gleich sein.

7.13.2 HRDCPY

Format: HRDCPY

Dieser Befehl führt eine Hardcopy eines LOW-RESOLUTION-Bildschirms aus. Man hat die Möglichkeit, Daten auf dem Bildschirm auszugeben und dann mit einem Befehl auf dem Drucker.

7.14 BCKGNDS

FORMAT: BCKGNDS bf,h1,h2,h3

ZIEL: Betimmen der Hintergrundfarbe eines Zeichens.

Jedes Zeichen auf dem Bildschirm besteht aus 8 x 8 Punkten.
Die Hintergrundfarbe dieses Quadrates entspricht normalerweise der Farbe des Bildschirms (ausgenommen bei reverser Darstellung).
Der Befehl BCKGNDS bestimmt die Hintergrundfarbe eines Zeichens

in normaler und reverser Schrift. Beachten Sie, daß dies nur für solche Zeichen möglich ist, die sich oben auf der Taste befinden. Grafik-Zeichen können nicht benutzt werden.

Der Parameter bf des "BCKGNDS"-Befehls legt die Bildschirmfarbe fest. Die nächsten drei Parameter bestimmen die Hintergrundfarbe eines "geshifteten" Zeichens, eines "ungeshifteten" Zeichens in reverser Darstellung und eines "geshifteten" Zeichens in reverser Darstellung.

BEISPIEL: 3 verschiedene Hintergrundfarben der Zeichen (die unterstrichenen Zeichen sind mit gedrückter SHIFT-Taste zu schreiben):

EINGABE:

```
10 PRINT"<SHIFT CLR/HOME>"
20 BCKGNDS 1,3,5,7
30 PRINT"THIS IS AN EXAMPLE": PRINT
40 PRINT"<CTRL RVS ON> OF THE SIMON'S BASIC": PRINT
50 PRINT"<CTRL RVS ON> BCKGNDS COMMOAND": PRINT

RUN <RETURN>
```

ANZEIGE: 3 Zeilen Text mit türkisem, grünem und gelbem Hintergrund.

ABSCHNITT ACHT

SPRITE UND GRAFIK

8.1 Einführung

SIMON's-BASIC bietet zwei grafische Besonderheiten. Zum Einen wird ermöglicht, einen anderen Zeichensatz zu erstellen und die Tasten des Commodore 64 anders zu belegen.

Jedes Zeichen wird dabei in einer 8x8 Matrix dargestellt. (s. Abschnitt 8.3).
Die zweite Besonderheit ist die Möglichkeit, 'SPRITES' zu erstellen und diese in Bewegung zu versetzen (s. CBM-Handbuch Seite 67). Dies geschieht jedoch beim Standard-BASIC durch viele POKE-Befehle. Bei SIMON's-BASIC werden diese POKE's durch einfach zu handhabende BASIC-Befehle ersetzt.
Ein SPRITE wird in SIMON's-BASIC 'moveable object block' oder MOB genannt. Es können bis zu acht unabhängige MOB's gleichzeitig dargestellt werden.
Zwei Arten von MOB's sind möglich:

- High-Resolution: 24x21 Matrix, einfarbig
- Multi-Colour: 12x21 Matrix, bis zu drei Farben

Sie können auf dem normalen Bildschirm oder im Grafik-Modus dargestellt werden.

8.2 Gestalten eines MOB's

8.2.1 DESIGN

Format: DESIGN c,ad
oder DESIGN c,sa + gc
Ziel: Zuteilung von Speicherplatz für einen MOB

Der DESIGN-Befehl reserviert den benötigten Speicherplatz im Speicher des CBM 64 für einen MOB, der konstruiert weden soll.
Der erste Parameter c gibt an, ob der MOB ein high-resolution (0) oder multi colour (1) MOB ist.
Der zweite Parameter ad definiert die Start-Adresse im Speicher des Computers. Diese Start-Adresse errechnet sich aus einer Block-Nummer multipliziert mit 64, da jeder MOB 64 Byte Speicherplatz benötigt.

Block-Nummer	Speicherplatz
13 - 15	832 - 1023
32 - 63	2048 - 4095
128 - 255	8192 - 16383

Wenn ein MOB auf einem high-resolution Bildschirm dargestellt werden soll, so muß zu der Startadresse ein konstanter Wert von 49152 ($C000) addiert werden.

Wird im Programm der Befehl MEM (s. Abschnitt 8.3.1) verwendet, so können nur Block 192 bis 255 für MOB's benutzt weden.

Achtung:
Es können so viele MOB's erstellt werden, wie freier Speicherplatz im CBM 64 zur Verfügung steht. Auf dem Bildschirm können jedoch nur 8 MOB's gleichzeitig dargestellt werden. Es ist möglich, in einem Programm einen MOB durch einen anderen zu ersetzen, in dem man unter der gleichen Startadresse einen neuen MOB entwirft.

Beispiel:
Speicherplatz bereitstellen für einen High-Resolution MOB auf einem normalen Bildschirm.

```
100 DESIGN 0,832
```

Ergebnis:
Der MOB wird in Block 13 (13x64=832) gespeichert.
Siehe auch Abschnitt 8.2.7

8.2.2 @ (Klammeraffe)

Format: @ ....
@ ....

Ziel : Die Form eines MOB's eingeben.

Ein MOB besteht, wie bereits gesagt, entweder aus einer Matrix

von 24x21 (High Resolution) oder 12x21 (Multi Colour) Punkten. Der Befehl @ bietet eine sehr einfache Möglichkeit, einen MOB zu gestalten.
Jeder MOB besteht aus 21 BASIC-Zeilen mit dem Format @ ...., wobei jede Zeile aus einem @-Zeichen und 24 (High Resolution) bzw. 12 (Multi Colour) Punkten besteht. Jeder Punkt stellt dabei einen Punkt der Matrix eines MOB's in der Farbe des Bildschirm-Hintergrunds dar. Ersetzt man diesen Punkt durch einen bestimmten Buchstaben, so ändert sich die Farbe dieses Punktes in der Matrix.

High Resolution MOB's

Buchstabe:		Farbe:
B	=	Farbe, die im Befehl MOB SET angegeben ist.

Multi Colour MOB's

Buchstabe:		Farbe:
B	=	Farbe 1 (c1), die im Befehl CMOB angegben ist.
C	=	Farbe, die im Befehl MOB SET angegeben ist.
D	=	Farbe 2 (c2), die im Befehl CMOB angegeben ist.

Beispiel: siehe 8.2.7

8.2.3 CMOB

Format: CMOB c1, c2
Ziel: Festlegen der Farben für multi-colour MOB's

Der CMOB-Befehl legt die beiden zusätzlichen Farben für alle Multi-Colour MOB's fest. Die Parameter c1 und c2 können Werte zwischen 0 und 15 annehmen und entsprechen den Farben der Tabelle aus Abschnitt 6.

Alle Punkte der Matrix, die durch den Buchstaben B dargestellt werden, nehmen die Farbe entsprechend dem Parameter c1 an; die Punkte, die durch ein D dargestellt werden, entsprechend dem Parameter c2.

8.2.4 MOB SET

Format: MOB SET mb,blk,col,pr,res
Ziel: Festlegen, wie der MOB aufgebaut ist.
Der Befehl MOB SET legt bestimmte Eigenschaften fest, die einen MOB bestimmen.

mb : Nummer des MOB's, kann Werte von 0 bis 7 annehmen. Wenn mehrere MOB's auf dem Bildschirm sind, bewegt sich der MOB mit der niedrigeren Nummer vor dem MOB mit der höheren Nummer.

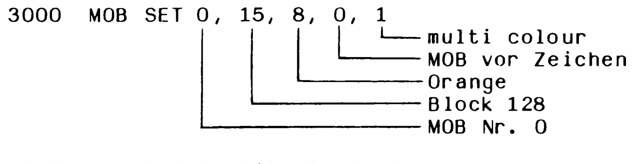
blk: legt fest, aus welchem Block die Daten für den MOB entnommen werden sollen (s. Abschnitt 8.2.1).

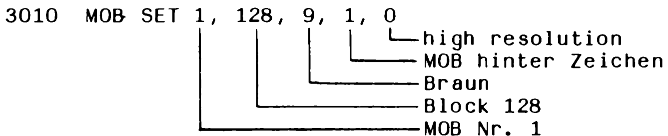
col: legt die Farbe für den betreffenden MOB fest. Der Parameter kann entsprechend einer Farbe Werte zwischen 0 und 15 annehmen.
Bedeutung bei High-Resolution MOB's:
Punkte der Matrix, die mit einem B (s. Abschnittt 8.2.2) gekennzeichnet sind, werden mit der festgelegten Farbe ausgefüllt.
Bedeutung bei Multi-Colour MOB's:
Punkt der Matrix, die mit einem C (s. Abschnitt 8.2.2) gekennzeichnet sind, werden mit der festgelegten Farbe ausgefüllt.

pr : legt die Priorität des MOB fest
pr= 0 : Der MOB bewegt sich vor den Bildschirmzeichen.
pr= 1 : Der MOB bewegt sich hinter den Bildschirmzeichen.

res: res= 0 high-resolution MOB
res= 1 multi colour MOB

Beispiel:





siehe auch 8.2.7

8.2.5 MMOB

Format: MMOB mn,x1,y1,x2,y2,exp,speed
Ziel: Darstellen und/oder Bewegen eines MOB's

Der Befehl MMOB bewirkt, daß ein MOB auf dem Bildschirm dargestellt wird und ermöglicht eine Bewegung dieses MOB's

mn : gibt die Nummer des MOB's an, der dargestellt werden soll

x1,y1: Start Koordinaten, von denen aus sich der MOB bewegt, bzw. dargestellt wird

x2,y2: Ziel-Koordinaten, auf die sich der MOB zubewegt

exp : gibt die Größe des MOB an

exp	Größe		
0	normale Größe (24x21)		
1	x-Achse ist gedehnt:	(48x21)	Faktor 2
2	y-Achse ist gedehnt:	(24x42)	Faktor 2
3	x- u. y-Achse ist gedehnt:	(48x42)	jeweils Faktor 2

speed: Geschwindigkeit, mit der sich der MOB bewegt (1 bis 255)

speed= 1: schnellste Geschwindigkeit
speed= 255: langsamste Geschwindigkeit

Beispiel: siehe Abschnitt 8.2.7

8.2.6 RLOCMOB

Format: RLOCMOB mn,x,y,exp,speed
Ziel: Bewegen eines MOB's

RLOCMOB bewegt einen auf dem Bildschirm dargestellten MOB zu einer anderen Position auf dem Bildschirm. Die Koordinaten x,y bestimmen die Zielposition. Alle anderen Parameter entsprechen den Parametern von Abschnitt 8.2.5.

Beispiel: siehe Abschnitt 8.2.7

8.2.7 MOB OFF

Format: MOB OFF mn
Ziel: Löschen eines MOB's vom Bildschirm

Der Parameter mn gibt die Nummer des MOB's an, der gelöscht werden sollen.

Beispiel siehe Seite 51-53

```
9900 REM *****************************
9910 REM *** BEISPIEL:             ***
9920 REM *** MOBS MIT SIMONS' BASIC ***
9930 REM *****************************
9940 REM *** ENTWURF EINES          ***
9950 REM *** HIGH-RESOLUTION-MOB    ***
9960 REM ***  IM BLOCK 13           ***
9970 REM *****************************
9980 PRINT"♥"
10000 DESIGN 0,13*64
10010 @..........B.B...........
10011 @...........BB.BB........
10012 @............BB..B.......
10013 @............BBBBB.......
10014 @...........BBBB.BBB.....
10015 @..........BBBBB....B....
10016 @.........BBBBB..........
10017 @.......BBBBBBBBBB.......
10018 @......BBBBBBBB..BB......
10019 @....BBBBBBBBBB..........
10020 @...BBBBBB..BB...........
10021 @..BB.BBBBBBB............
10022 @..BB.BBBBBBBB...........
10023 @.BB...BBBBBBB...........
10024 @.BB.....BBBBBB..........
10025 @BB.........BBBB.........
10026 @BB.......BBBB...........
10027 @........BBB.............
10028 @.......BB...............
10029 @......BBBBBBB...........
10030 @.......BBBBBBB..........
10035 REM *****************************
10036 REM *** ENTWURF EINES         ***
10037 REM *** HIGH-RESOLUTION-MOB   ***
10038 REM ***  IN BLOCK 14          ***
10039 REM *****************************
10040 DESIGN 0,14*64
10050 @.............BB.........
10051 @.............BBBB.BBB...
10052 @..............BBBBBBBB..
10053 @................BBBB.BB.
10054 @...............BBBBBBBBB
10055 @..............BBBBBB....
10056 @.............BBBBBB.....
10057 @..........BBBBBBBBBBBBB.
10058 @.........BBBBBBBBBBBBBBB
10059 @.......BBBBBBBBBBBB...BB
10060 @......BBBBBBBBBBBBB.....
10061 @......BBBBBBB..BBBB.....
10062 @.....BBBBBBBBBB.BBB.....
10063 @....BBBBBBBBBBB.BBBB....
10064 @....BBBBBBBBBB.BBBB.....
10065 @...BBBB.BBBBBBBBBB......
10066 @..BBBB...BBBBBBB........
10067 @BBBBB....BBBB...........
10068 @BBB.....BBBB...BBB......
10069 @........BBBBBBBBBBB.....
10070 @.........BBBBBBBBB......
```

```
10075 REM ******************************
10076 REM *** ENTWURF EINES          ***
10077 REM *** MULTI-COLOUR-MOB       ***
10078 REM *** IN BLOCK 15            ***
10079 REM ******************************
10080 DESIGN 1,15*64
10090 @.....BB.....
10091 @..BBBBCCCB..
10092 @.BBBBCCCCCB.
10093 @.BBBCCCCCCC.
10094 @BBBBCCCDDCCB
10095 @BBBBCCCDDCCB
10096 @.BBBCCCCCCC.
10097 @.BBBBCCCCCB.
10098 @..BBBBCCCB..
10099 @.D...BB...D.
10100 @............
10101 @..D......D..
10102 @............
10103 @...D....D...
10104 @............
10105 @....D..D....
10106 @....CCCC....
10107 @....CCCC....
10108 @....CCCC....
10109 @...DCCCCD...
10110 @...DDDDDD...
11990 REM ******************************
11991 REM *** MULTI-COLOUR-MOB NR.1 ***
11992 REM *** EXPANSION IN Y-RICHT. ***
11993 REM *** BEWEG. 0/0-300/205    ***
11994 REM ******************************
12000 CMOB 7,2
12010 MOB SET 1,15, 8,0,1
12020 MMOB 1,  0,  0,300,205,2,200
12040 REM ******************************
12041 REM *** MULTI-COLOUR-MOB NR.2 ***
12042 REM *** EXPANSION IN X U. Y   ***
12043 REM *** BEWEG. 200/200-300/50 ***
12044 REM ******************************
12050 CMOB 0,14
12060 MOB SET 2,15,6,1,1
12070 MMOB 2,200,200,300,50,3,150
12090 REM ******************************
12091 REM *** MULTI-COLOUR-MOB NR.7 ***
12092 REM *** EXPANSION IN X-RICHT. ***
12093 REM *** BEWEG. 100/0-300/150  ***
12094 REM ******************************
12100 CMOB 8,5
12110 MOB SET 7,15,13,0,1
12120 MMOB 7,100,0,300,150,1,100
12125 REM ******************************
12126 REM *** MULTI-COLOUR-MOB NR.6 ***
12127 REM *** KEINE EXPANSION       ***
12128 REM *** BEWEG. 10/150-300/175 ***
12129 REM ******************************
12130 CMOB 3,4
12140 MOB SET 6,15,0 ,1,1
12150 MMOB 6,10,150,300,175,0,50
```

```
12157 REM *****************************
12158 REM *** NEUE FARBE            ***
12159 REM *****************************
12160 CMOB 7,2
14990 PRINT"
14995 REM *****************************
14996 REM *** HIGH-RES.-MOB NR.5    ***
14997 REM *** AUS BLOCK 14          ***
14998 REM *****************************
15000 MOB SET 5,14,6,0,0
15995 REM *****************************
15996 REM *** HIGH-RES.-MOB NR 4    ***
15997 REM *** AUS BLOCK 13          ***
15998 REM *****************************
16000 MOB SET 4,13,3,5,0
19990 REM *****************************
19991 REM *** GLEICHZEITIGE BEWEGUNG  *
19992 REM *** VON MOB 4 UND 5.        *
19993 REM *** EXPANSION VON MOB 5 IN  *
19994 REM *** X- U. Y-RICHTUNG.       *
19995 REM *****************************
20000 Y=150 :Z=50
20010 FOR X=30 TO 280 STEP 4
20020 Y=Y-2:Z=Z+2
20022 IF Y<60 THEN Y=60
20023 IF Z>140 THEN Z=140
20030 FOR DY=1 TO 10
20040 RLOCMOB 5,X,Y+DY*4,3,0
20045 RLOCMOB 4,X,Z+DY*2,0,1
20050 NEXT DY
20060 NEXT X
20090 REM *****************************
20091 REM *** MOB LOESCHEN          ***
20092 REM *****************************
20100 PRINT"MOB LOESCHEN (J ODER N)?"
20110 FETCH"JN",1,ANT$
20120 IF ANT$="N" THEN 20160
20130 FOR L=0 TO 7
20140 :  MOB OFF L
20150 NEXT L
20160 END
```

8.2.8 DETECT

Format: DETECT n
Ziel: Abfrage der MOB-Kollision vorbereiten

Der DETECT-Befehl muß vor dem Befehl CHECK im Programm kommen. Ist der Parameter n=0, so wird die Abfrage auf eine Kollision zwischen zwei MOB's vorbereitet. Ist n=1, so wird die Kollision zwischen Zeichen auf dem Bildschirm und einem MOB vorbereitet. Der Befehl DETECT muß zweimal gegeben werden. Bei der ersten Ausführung wird das SPRITE-KOLLISIONS-REGISTER gelöscht; bei der zweiten Ausführung ist das Register bereit, eine Kollision zu registrieren.

8.2.9 CHECK

FORMAT: IF CHECK(mn1, mn2)=0 THEN Anweisung
oder: IF CHECK(0)=0 THEN Anweisung
Ziel: Abfrage auf MOB-Kollision

Der Befehl CHECK fragt ab, ob eine Kollision stattgefunden hat. Die Parameter mn1 und mn2 geben an, zwischen welchen beiden MOB's die Abfrage gemacht werden soll. Steht in den Klammern eine "0", dann wird abgefragt, ob ein MOB mit einem Bildschirmzeichen kollidiert ist. Hat eine Kollision stattgefunden, so wird die Anweisung hinter THEN ausgeführt.

Beispiel:
Einfügen in "MOB's mit SIMON's-BASIC":

```
20047 DETECT 0 : DETECT 0
20048 IF CHECK(4,5)=0 THEN END
```

8.3 Erstellen eines neuen Zeichensatzes

8.3.1 MEM

Format: MEM
Ziel: Verlegen des Zeichensatzes vom ROM- in den RAM-Bereich

Der Zeichensatz des Commodore 64 ist in einem ROM festgelegt. Um diesen Zeichensatz neu zu definieren oder zu verändern, muß man ihn in den RAM-Bereich legen.
Der Befehl MEM führt folgende Funktionen aus:
- Verschieben des Zeichensatz-ROM-Inhalts in den RAM-Bereich hinter KERNAL
- Verschieben des Bildschirm-RAM nach $ CC00
- Eingrenzen des MOB-Bereiches auf Block 192-255 (8.2.1)

8.3.2 DESIGN

Format: DESIGN 2, $E000 + ch x 8
Ziel: Festlegen, welches Zeichen von dem neu erstellten Zeichen ersetzt wird.

Der DESIGN-Befehl legt fest, welches Zeichen durch ein neu definiertes Zeichen ersetzt wird. Der Parameter ch muß dabei den POKE-Code des zu ersetzenden Zeichens annehmen (s. Handbuch Seite 133/134). Das neu definierte Zeichen besteht aus einer Matrix von 8x8 Punkten. Wie die Matrix festgelegt wird, ist im folgenden Abschnitt erklärt.

8.3.3 @ (Klammeraffe)

Format: @ ....

Der Befehl @ ermöglicht die einfache Gestaltung eines neuen Zeichens. Nach dem @ -Zeichen setzt man 8 Punkte. Da ein Zeichen aus einer Matrix von 8x8 Punkten besteht, sind 8 BASIC-Zeilen des Befehls @ ... nötig, um ein neues Zeichen zu definieren. Jeder Punkt entspricht dabei einem Punkt der Matrix in der Bildschirm-Hintergrundfarbe. Soll ein Punkt der Matrix die Farbe des Cursors annehmen, so muß der Punkt durch ein B ersetzt werden.

Beispiel:

```
100 REM *********************
110 REM *** BEISPIEL:     ***
120 REM *** NEUES ZEICHEN ***
130 REM *********************
150 MEM
170 REM *** POKE-CODE 0 AENDERN ***
190 DESIGN 2,$E000
200 @.BBBBBB.
210 @.BB...B.
220 @..BB....
230 @...BB...
240 @..BB....
250 @.BB...B.
260 @.BBBBBB.
270 @........
```

Um wieder in den Originalzeichensatz zu gelangen, muß die RESTORE -Taste zusammen mit der RUN/STOP -Taste gedrückt werden.

ABSCHNITT NEUN

STRUKTURIERTE PROGRAMMIERUNG

9.1 EINFÜHRUNG

Ein großes Problem beim Programmieren im Standard-BASIC ist das Fehlen eines strukturierten Verlaufs bei komplexeren Programmen. Die Befehle GOTO und GOSUB machen die Programmlistings unübersichtlich und unverständlich. Etwas Abhilfe schaffen da Kommentarzeilen (s. REMark-Befehl), in denen die einzelnen Routinen erklärt weden.
SIMON's-BASIC beseitigt diese Probleme mit der Schaffung bestimmter Befehle zur strukturierten Programmierung. Diese Befehle verhindern den Gebrauch von GOTO und GOSUB weitgehend.

9.2 Bedingungen Prüfen und Programmschleifen

9.2.1 IF... THEN...ELSE

Format: IF Bedingung THEN Anweisung für wahr: ELSE: Anweisung für unwahr
Ziel: Auf eine Bedingung prüfen und je nach "wahr" und "unwahr" eine Anweisung ausführen.

Dieser Befehl ist im Prinzip mit dem Befehl IF... THEN... vergleichbar. Der Unterschied besteht darin, daß man zusätzlich die Möglichkeit hat, eine Anweisung zu geben, die ausgeführt wird, wenn die Bedingung nicht erfüllt ist.

Achtung:
ELSE muß durch Doppelpunkte (:) vom vorhergehenden und nachfolgenden Zeichen abgetrennt werden.

Beispiel:

```
100 REM ***************************
110 REM *** BEISPIEL:            ***
120 REM *** IF...THEN...ELSE...  ***
130 REM ***************************
140 PRINT"◣"
150 PRINT"ENDE DES PROGRAMMS (J ODER N) ?"
160 FETCH"JN",1,ANTW$
170 REM *** ABFRAGE JA,NEIN      ***
180 IF ANTW$="J" THEN END:ELSE:GOTO 150
190 END
```

9.2.2 REPEAT... UNTIL

Format: REPEAT: Schleife : UNTIL Bedingung erfüllt
Ziel: Wiederholen einer Anweisung bis eine Bedingung erfüllt ist. REPEAT... UNTIL führt eine ähnliche Funktion wie eine FOR...NEXT-Schleife im Standard-BASIC aus. REPEAT startet die Ausführung der Schleife. UNTIL prüft auf eine Bedingung. Die Schleife wird verlassen, wenn die Bedingung erfüllt ist.

Beispiel:

```
100 REM ****************************
110 REM *** BEISPIEL             ***
120 REM *** REPEAT...UNTIL...    ***
130 REM ****************************
150 A=65
160 REM *** SCHLEIFENANFANG      ***
170 :  REPEAT
180 :     PRINT CHR$(A)
190 :     A=A+1
200 :  UNTIL A>71
210 REM *** SCHLEIFENENDE        ***
220 PRINT"ENDE"
```

Ergebnis:
Es werden die Zeichen A-G auf den Bildschirm geschrieben.

9.2.3 RCOMP

Format: RCOMP: Anweisung für wahr: ELSE: Anweisung für unwahr
Ziel: Abfrage nach der gleichen Bedingung des zuletzt ausgeführten IF...THEN...ELSE-Befehls

RCOMP ermöglicht es, die Bedingung in der zuletzt ausgeführten IF...THEN...ELSE-Abfrage zu prüfen.
Dies ist besonders dann eine Erleichterung, wenn die Bedingung aus mehreren Verknüpfungen besteht.

Beispiel:

```
100 REM ****************************
110 REM *** BEISPIEL :           ***
120 REM *** RCOMP                ***
130 REM ****************************
150 INPUT A
160 IF A=10 THEN PRINT "HALLO "; : ELSE:      PRINT "AUF WIEDERSEHN ";
170 RCOMP:PRINT"HANS ";:ELSE:PRINT"PETER ";
180 RCOMP:PRINT"UND RENATE.":ELSE:PRINT"UND ERICH."
190 END
```

Ergebnis:
Nach dem INPUT wird dreimal auf die Bedingung A=10 abgefragt.

9.2.4 LOOP...EXIT IF... END LOOP

Format: LOOP:Programmschleife:EXIT IF Bedingung:END LOOP
Ziel: Durchlaufen einer Schleife, bis eine Bedingung erfüllt ist.

Dieser Befehl ermöglicht es, eine Programmschleife aufzubauen, in der die Bedingung des Schleifenendes in der Schleife festgelegt ist. Es können mehrere Bedingungen in der Schleife vorkommen. Ist eine Bedingung erfüllt, dann wird die Schleife verlassen, und der Befehl hinter der Programmschleife ausgeführt. Ist keine Bedingung erfüllt, so wird die Schleife erneut durchlaufen.

Beispiel:
Zeichen zwischen A und F von der Tastatur auf den Bildschirm bringen.

```
100 REM ********************************
110 REM *** BEISPIEL                 ***
120 REM *** LOOP...                  ***
130 REM ***   EXIT IF...             ***
140 REM *** END LOOP...              ***
160 REM ********************************
170 PRINT"BITTE EINEN BUCHSTABEN ZWISCHEN A UND F EINGEBEN!"
180 REM *** SCHLEIFENANFANG          ***
190 LOOP
200 :  GET A$
210 :  IF A$="" THEN 200
220 :  EXIT IF ASC(A$) < 66
230 :  EXIT IF ASC(A$) > 69
240 :  PRINT A$
250 END LOOP
260 REM *** SCHLEIFENENDE            ***
270 PRINT A$;" LIEGT NICHT ZWISCHEN A UND F!"
280 END
```

9.3 Programmiertechnik

9.3.1 Einführung

Um das Schreiben strukturierter Programme zu erleichtern, bietet SIMON's-BASIC vier Befehle, die es erlauben, Programm-Routinen mit Namen (LABEL) zu versehen und diese darunter aufzurufen. Der Gebrauch von GOTO und GOSUB wird zum großen Teil überflüssig.
In den folgenden Abschnitten werden diese Befehle erklärt und ein umfassendes Beispiel gegeben.

9.3.2 PROC

Format: PROC Label
Ziel: Vergeben einer symbolischen Sprungadresse (Label)

PROC bietet die Möglichkeit, einer Programmroutine eine symbolische Sprungadresse (Label) zuzuweisen. Alle Zeichen hinter dem Befehl PROC bilden den Label, unter dem die Routine dann aufgerufen werden kann. In dieser Programmzeile dürfen keine weiteren Anwesungen vorkommen.

Beispiel:
Einer Programmroutine den Label "INPUT NAME" zuweisen.
1000 PROC INPUT NAME

9.3.3 END PROC

Format: END PROC
Ziel: Das Ende einer Routine festlegen.

END PROC legt das Ende einer geschlossenen Routine fest. Der Befehl ist mit dem RETURN des Standard-BASIC vergleichbar. Er bewirkt den Rücksprung auf den unmittelbar hinter EXEC folgenden Befehl.

Beispiel:

```
400 PROC INPUT NAME
450 PRINT"■BITTE EINEN NAMEN EINGEBEN !"
460 FETCH" ■",20,A$
470 REM *** ENDE DES UNTERPROGRAMMS***
480 END PROC
```

Ergebnis:
Dieses Unterprogramm kann mit dem Befehl EXEC INPUT NAME aufgerufen werden.

Achtung:
END PROC darf nur hinter EXEC ausgeführt werden, da sonst die Fehlermeldung
? END PROC WITHOUT EXEX IN... kommt.

9.3.4 CALL

Format: CALL Label
Ziel: Sprung zu der symbolischen Sprungadresse (Label).

Der Befehl CALL entspricht dem Befehl GOTO des Standard-BASIC. Hier wird aber nicht in eine feste Zeilennummer gesprungen, sondern zu einer symbolischen Sprungadresse (Label).

9.3.5 EXEC

Format: EXEC Label
Ziel: Unterprogrammaufruf

Der EXEC-Befehl entspricht dem GOSUB-Befehl des Standard-BASIC. Das Unterprogramm wird hier aber unter einem symbolischen Namen (Label) aufgerufen. Nach dem Rücksprung aus dem Unterprogramm wird der Befehl unmittelbar hinter EXEC ausgeführt.

Beispiel:

```
100 REM ******************************
110 REM *** BEISPIEL:              ***
120 REM *** PROC...END PROC        ***
130 REM *** CALL...EXEC...         ***
140 REM ******************************
150 REM *** HAUPTPROGRAMM:         ***
160 REM *** SORTIERPROGRAMM FUER NAMEN*
170 REM ******************************
180 I=1
190 PRINT"■****  SORTIERPROGRAMM FUER NAMEN  ****"
200 PRINT"■SIE KOENNEN HIER 10 NAMEN EINGEBEN, DIE DANN ALPHABETISCH SORTIERT ";
210 PRINT"WERDEN."
220 PRINT"■DIE NAMEN DUERFEN NICHT LAENGER ALS 20  BUCHSTABEN SEIN."
230 PROC WEITER
240 EXEC INPUT NAME
250 A$(I)=A$
260 IF I<10 THEN I=I+1:ELSE:CALL FERTIG
270 PRINT"■NOCH EINEN NAMEN (J ODER N) ?"
280 FETCH"JN",1,B$
290 IF B$="J" THEN CALL WEITER
300 I=I-1
310 PROC FERTIG
320 IF I>1 THEN EXEC SORT
330 PRINT"■"
340 FOR A=1 TO I
350 : Z$=STR$(A)
360 : USE"##. NAME : ",Z$:PRINT A$(A)
370 NEXT A
380 END
390 :
```

```
400 PROC INPUT NAME
410 REM ******************************
420 REM *** UNTERPROGRAMM:         ***
430 REM *** EINGABE DES NAMEN      ***
440 REM ******************************
450 PRINT"■BITTE EINEN NAMEN EINGEBEN !"
460 FETCH" ■",20,A$
470 REM *** ENDE DES UNTERPROGRAMMS***
480 END PROC
490 :
500 PROC SORT
510 REM ******************************
520 REM *** UNTERPROGRAMM:         ***
530 REM *** SORTIEREN VON NAMEN    ***
540 REM ******************************
550 REPEAT
560 :   T=0
570 :   FOR A=1 TO I-1
580 :    IF A$(A)<A$(A+1) THEN CALL NEXT
590 :    IF A$(A)=A$(A+1) THEN CALL NEXT
600 :    W$=A$(A) : A$(A)=A$(A+1) : A$(A+1)=W$ : T=1
610 :    PROC NEXT
620 :   NEXT A
630 UNTIL T=0
640 REM *** ENDE DES UNTERPROGRAMMS***
650 END PROC
```

```
****  SORTIERPROGRAMM FUER NAMEN  ****
SIE KOENNEN HIER 10 NAMEN EINGEBEN, DIE DANN ALPHABETISCH SORTIERT WERDEN.
DIE NAMEN DUERFEN NICHT LAENGER ALS 20  BUCHSTABEN SEIN.
BITTE EINEN NAMEN EINGEBEN !
UWE
NOCH EINEN NAMEN (J ODER N) ?
J
BITTE EINEN NAMEN EINGEBEN !
UTE
NOCH EINEN NAMEN (J ODER N) ?
J
BITTE EINEN NAMEN EINGEBEN !
KARL
NOCH EINEN NAMEN (J ODER N) ?
J
BITTE EINEN NAMEN EINGEBEN !
KARLCHEN
NOCH EINEN NAMEN (J ODER N) ?
N

 1. NAME : KARL
 2. NAME : KARLCHEN
 3. NAME : UTE
 4. NAME : UWE
```

9.4 VARIABLE

9.4.1 Einführung

Im Standard-BASIC müssen alle Variablen einen bestimmten Wert haben. Im SIMON's-BASIC besteht nun die Möglichkeit, einer Variablen einen globalen Wert für das ganze Programm und an bestimmten Stellen einen weiteren lokalen Wert zuzuweisen. Dabei geht der globale Wert nicht verloren.

9.4.2 LOCAL

Format: LOCAL Variable 1, Variable 2, Variable 3, ...
Ziel : Festlegen der Variablen an einer bestimmten Stelle im Programm.

Der LOCAL-Befehl weist den Variablen neue Werte zu. Der ursprüngliche Wert dieser Variablen bleibt erhalten. Im Abschnitt 9.4.3 wird der Befehl LOCAL in einem Beispiel angewendet.

9.4.3 GLOBAL

Format: GLOBAL
Ziel : Wiederherstellen der ursprünglichen Wertzuweisung im Programm.

Der GLOBAL-Befehl stellt den ursprünglichen Zustand aller Variablen wieder her, die mit dem Befehl LOCAL verändert wurden.

Beispiel:

```
100 REM ******************************
110 REM *** BEISPIEL:              ***
120 REM *** GLOBAL... LOCAL...     ***
130 REM ******************************
140 PRINT"␣"
150 REM *** FESTLEGEN DER          ***
160 REM *** GLOBALEN WERTE         ***
170 A$="GLOBALER WERT":A%=123:A=456.7
180 REM *** FESTLEGEN DER          ***
190 REM *** NEUEN WERTE            ***
200 LOCAL A$,A%,A
210 A$="NEUER WERT":A%=456:A=89.1
220 PRINT A$,A%,A
230 REM *** WIEDERHERSTELLEN DER   ***
240 REM *** GLOBALEN WERTE         ***
250 GLOBAL
260 PRINT A$,A%,A
270 END
```

Ergebnis:

Unter den neuen Werten werden die ursprünglichen Werte angezeigt.

9.5 FEHLER-BEHANDLUNG

9.5.1 EINFÜHRUNG

SIMON's-BASIC bietet die Möglichkeit, Fehler im Programm aufzufangen und das Abstürzen zu verhindern. Tritt ein Fehler auf, dann wird zu einer bestimmten Stelle im Programm verzweigt. Mit den Variablen ERRN und ERRLN kann anschließend festgestellt werden, um welchen Fehler (ERRN) es sich handelt, und in welcher BASIC-Zeile (ERRLN) er sich befindet. Mit diesen Informationen können Fehlermeldungen mit eigenem Text erstellt werden. Mit NO ERROR bzw. OUT kehrt man zur Fehlermeldung des Commodore 64 zurück.

9.5.2 ON ERROR

Format: ON ERROR : GOTO Zeilennummer
Ziel : Abfangen eines Programmfehlers
Wie in einem Beispiel des Abschnitts 9.5.3 gezeigt wird, kann der Absturz eines Programms mit Hilfe dieses Befehls verhindert werden. Tritt ein Fehler auf, so springt das Programm in die angegebene Zeilennummer. In der Variablen ERRN wird die Fehlernummer und in ERRLN die Zeilennummer, in der sich der Fehler befindet, gespeichert.
Folgende Fehler können mit SIMON's-BASIC aufgefangen werden:

Fehlernummer	Fehler
1	TOO MANY FILES
2	FILE OPEN
3	FILE NOT OPEN
4	FILE NOT FOUND
5	DEVICE NOT PRESENT
10	NEXT WITHOUT FOR
11	SYNTAX
12	RETURN WITHOUT GOSUB
13	OUT OF DATA
14	ILLEGAL QUANTITY
15	OVERFLOW
16	OUT OF MEMORY
17	UNDEFINED STATMENT
18	BAD SUBSCRIPT
19	RE-DIMENSIONED ARRAY
20	DIVISION BY ZERO
21	ILLEGAL DIRECT
22	TYPE MISMATCH
23	STRING TOO LONG

9.5.3 NO ERROR

Format: NO ERROR
Ziel : Fehlermeldung unterdrücken.
Mit dem Befehl NO ERROR wird eine Fehlermeldung unterdrückt. Die Fehlerbehandlung des ON ERROR-Befehls wird gleichzeitig abgebrochen. Das Programm fährt fort. Tritt ein weiterer Fehler auf, so kommt eine Fehlermeldung des Standard-BASIC.

Beispiel:

```
100 REM *******************************
110 REM *** BEISPIEL:              ***
120 REM *** FEHLERBEHANDLUNG       ***
130 REM *******************************
140 ON ERROR:GOTO 1000
150 REM *** TEST FEHLER            ***
160 PRIN "*********  TEST FEHLER  **************"
170 READ B
180 PRINT B: GOTO 170
190 DATA 1,2,3,4,5
200 PRINT"*********    TESTENDE    **************"
210 REM *** TESTENDE               ***
220 NO ERROR
230 END
1000 IF ERRN =11 THEN PRINT"SYNTAX-FEHLER IN ZEILE"; ERRLN:CGOTO(ERRLN+10)
1010 IF ERRN =13 THEN PRINT"ZU WENIG DATEN: FEHLER IN "; ERRLN:CGOTO(ERRLN+20)
1020 NO ERROR
1030 END
```

Nach dem Programmstart wird folgendes ausgedruckt:

```
SYNTAX-FEHLER IN ZEILE 160
 1
 2
 3
 4
 5
ZU WENIG DATEN: FEHLER IN  170
*********    TESTENDE    **************
```

Ergebnis:
Trotz der Fehler wird das Programm bis zum Ende ausgeführt.
Die neuen Fehlermeldungen erscheinen auf dem Bildschirm.

9.5.4 OUT

Format: OUT
Ziel : Abschalten der SIMON's-BASIC Fehlerbehandlung.

Nach dem Befehl OUT unterliegt die Kontrolle der Fehlerbehandlung wieder dem Commodore 64, d.h. es erscheinen die üblichen Fehlermeldungen des Standard-BASIC.

ABSCHNITT ZEHN

MUSIKMACHEN MIT SIMON'S-BASIC

10.1 Einleitung

Eine der interessantesten Besonderheiten des Commodore 64 ist die Möglichkeit, Musikstücke zu programmieren. Mit Übung und Erfahrung kann man den Klang vieler verschiedener Musikinstrumente nachbilden. Im Standard-BASIC benötigt man dazu viele POKE-Befehle. SIMON's-BASIC bietet Befehle, die die Musikprogrammierung wesentlich vereinfachen. Um mit dem Commodore 64 Musik zu machen, ist die Einstellung folgender Parameter erforderlich:

-Lautstärke
-Wellenform
-Hüllkurve
-Stimme
-Note

10.2 MUSIK-Befehle

10.2.1 VOL

Format: VOL n
Ziel : Lautstärke einstellen.

Der VOL-Befehl stellt die Lautstärke der drei Tongeneratoren ein. Der Parameter n kann Werte von 0 bis 15 annehmen.
n=0 schaltet die Tongeneratoren ab.
n=15 ist die maximale Lautstärke.

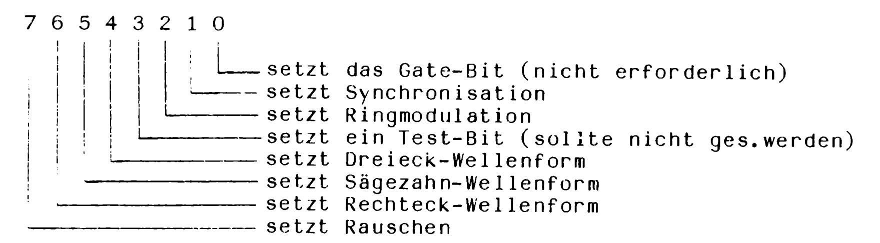
10.2.2 WAVE

Format: WAVE, St, Binärziffer
Ziel : Einstellen der Wellenform

Die Wellenform bestimmt den Klangcharakter der drei Tongeneratoren (Stimmen) des Commodore 64

St = 1	Stimme 1
St = 2	Stimme 2
St = 3	Stimme 3

Der zweite Parameter im WAVE-Befehl ist eine Binärziffer. Durch setzen der einzelnen Bits (1) wird der Charakter des Tones eingestellt.



Genauere Erklärung der einzelnen Bits:

Bit 0 - Das Gate-Bit

Dieses Bit triggert den Hüllkurven-Generator. Wenn dieses Bit gesetzt ist sind die ATTACK-DECAY-SUSTAIN-Cyclen initialisiert. Geht das Bit auf 0, dann beginnt der RELEASE-Cyclus. (siehe Abschnitt 10.4.3. und 10.4.4)
Dieses Bit wird automatisch bei der Ausführung des Play-Befehls gesteuert.

Bit 1 - Synchronisation

Dieses Bit gibt die Möglichkeit, die Frequenz einer Stimme mit der Frequenz einer anderen Stimme zu synchronisieren.

Bit 1 gesetzt in Stimme	Funktion
1	Synchronisiert Grundfrequenz Stimme 1 mit Stimme 3
2	Synchronisiert Grundfrequenz Stimme 2 mit Stimme 1
3	Synchronisiert Grundfrequenz Stimme 3 mit Stimme 2

Synchronisiert man eine Stimme mit variabler Frequenz mit einer, die eine feste Frequenz besitzt, so kann man sehr komplexe Klänge erzeugen. Die besten Ergebnisse erzielt man, wenn man die feste Frequenz tiefer wählt, als die variable Frequenz.

Bit 2 - Ringmodulation

Bei der Ringmodulation wird die Dreieckswelle durch eine Wellenform ersetzt, die durch Kombination der gewählten Stimme mit einer anderen (vom Betriebssystem festgelegten) Stimme entsteht. Auch hier empfiehlt sich (wie bei der Synchronisation), die Frequenz einer Stimme konstant zu halten und die andere zu variieren. Welche Stimmen kombiniert werden, gibt die folgende Tabelle an.

Stimme (Ausgang Dreieck) Bit 2 gesetzt	Modulations Signal
1	Stimme 1 mit Stimme 3
2	Stimme 2. mit Stimme 1
3	Stimme 3 mit Stimme 2

Hierbei entstehen nicht-harmonische Obertonschwingungen. Es können Spezialeffekte wie Gong und Glocken nachgebildet werden.

Bit 3 - Test Bit

Dieses Bit wird nicht in SIMON's-BASIC verwendet. Es sollte nicht gesetzt werde.

Bit 4 - Dreieck Wellenform

Schaltet den Dreieck-Oszillator ein. Eine Dreieck-Schwingung hat wenig harmonische Obertöne.

Bit 5 - Sägezahn Wellenform

Schaltet den Sägezahn-Oszillator ein. Eine Sägezahn-Schwingung hat viele gerade und ungerade harmonische Obertöne.

Bit 6 - Rechteck Wellenform

Schaltet den Rechteck-Oszillator ein. Der Anteil der Obertöne wird durch die Pulsbreite bestimmt.

Bit 7 - Rauschen

Schaltet den Rauschgenerator ein. Das Rauschen ist von den der Oszillatorfrequenz abhängig.

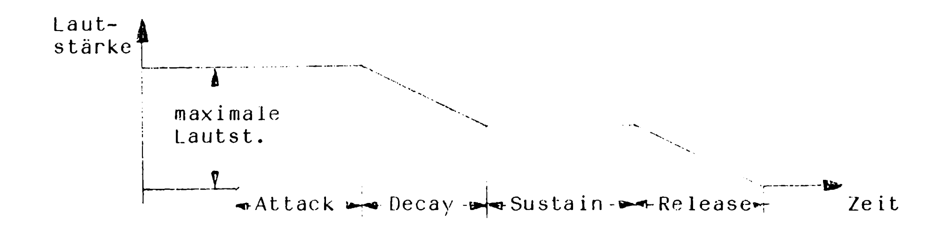
Beispiel.

```
20 WAVE 1,00010000
```

10.2.3 ENVELOPE

Format: ENVELOPE Stimme, Attack, Decay, Sustain, Release
Ziel : Einstellen der Hüllkurve.
Die Hüllkurve jeder Stimme kann getrennt eingestellt werden.
Sie besteht aus vier Parametern.
Attack, Decay, Sustain, Release

Bedeutung der Parameter:



Beispiel:
Erstellen einer Hüllkurve für Stimme 1

```
30 ENVELOPE 1,8,8,8,0
```

Ergebnis:
Alle Noten der Stimme 1, die nach diesem Befehl gespielt werden, folgen dem eingestellten Hüllkurvenverlauf.

10.2.4 MUSIC

Format: MUSIC n, "Noten"
oder : MUSIC n, Stringvariable
Ziel : Festlegen der Noten, die gespielt werden sollen.

In dem Befehl MUSIK werden weitere Parameter festgelegt, die den entstehenden Ton beeinflussen.
Der Parameter n bestimmt die Tondauer und muß im Bereich von 1 bis 255 liegen.
Der nächste Parameter legt die Tonhöhe und -dauer einer oder mehrerer Noten fest.
Zunächst wird in dem String nach SHIFT CLR/HOME die Stimme (1-3) festgelegt.
Danach folgt eine Steuertaste (F1-F8) für den Takt und die Note (C0-A7).

Funktionen der Steuertasten:

Taste	Funktion
SHIFT CLR/HOME	+ Ziffer 1-3 legt die Stimme fest.
F1	die nachfolgende Note wird als 1/16 Note gesp.
F3	" 1/8 "
F5	" 1/4 "
F7	" 1/2 "
F2	" 1/1 "
F4	" 2/1 "
F6	" 4/1 "
F8	" 8/1 "

Jede Note besteht aus einem Buchstaben (C-H) und einer Oktave (0-7). Man erhält die um einen halben Ton erhöhte Note, wenn man gleichzeitig mit dem Notennamen die SHIFT-Taste drückt. Am Ende jedes Noten-Strings muß SHIFT CLR/HOME und G eingegeben werden. Dies startet den RELEASE-Abschnitt der letzten Note.

Beispiel: siehe Abschnitt 10.2.5

10.2.5 PLAY

Format: PLAY n
Ziel : Wiedergabe der Musik

Der PLAY-Befehl startet das Spielen der komponierten Musik. Der Parameter n bezieht sich auf die Ausführung des weiteren Programms.

n=0 : Musik aus

n=1 : spielt die Musik und wartet mit der weiteren Ausführung des Programms.

n=2 : spielt Musik und fährt mit der Programmausführung fort.

Beispiel:

```
100 REM *******************************
110 REM *** BEISPIEL:                ***
120 REM *** MUSIK                    ***
130 REM *******************************
140 REM *** LAUTSTAERKE EINSTELLEN   ***
150 VOL 10
160 REM *** WELLENFORM EINSTELLEN    ***
170 WAVE 1,00010000
180 REM *** HUELLKURVE EINSTELLEN    ***
190 ENVELOPE 1,8,8,8,0
200 REM *** FESTLEGEN DER NOTEN      ***
210 A$= "■1Z■C5■E5■F5■"
220 A2$="G5■C5■E5■F5■G5■C5■E5■F5■G5■"
230 A2$=A2$+"E5■C5■E5■D5■E5■E5■D5■C5■"
240 A2$=A2$+"C5■E5■G5■G5■F5■F5■E5■F5■"
250 A2$=A2$+"G5■E5■C5■D5■C5■C5■C5■E5■"
260 A2$=A2$+"F5■"
270 A3$="C5■C5■"
280 MUSIC 8,A$+A2$+A2$+A3$
290 REM *** MUSIK SPIELEN            ***
300 PLAY 1
310 REM *** LAUTSTAERKE 0            ***
320 VOL 0
330 END
```

Ergebnis:

"WHEN THE SAINTS GO MARCHING IN"

wird gespielt.

ABSCHNITT ELF

FUNKTIONEN FÜR LIGHTPEN, JOYSTICK, ODER PADDLE

11.1 Einführung

Dieser Abschnitt beschreibt vier Lesefunktionen des SIMON's-BASIC. Mit diesen Funktionen kann die Position von LIGHTPEN, JOYSTICK oder PADDLE bestimmt werden.

11.2 PENX

Format: Variable = PENX
Ziel : Bestimmen der X-Koordinate des LIGHTPEN.

PENX gibt die Position des LIGHTPEN relativ zum linken Bildschirmrand an (0 bis 320)

11.3 PENY

Format: Variable = PENY
Ziel : Bestimmen der Y-Koordinate des LIGHTPEN.

PENY gibt den Abstand vom oberen Bildschirmrand an. (0 bis 200)

11.4 POT

Format: Variable = POT (0)
oder : Variable = POT (1)
Ziel : Den Widerstand des Paddle feststellen.

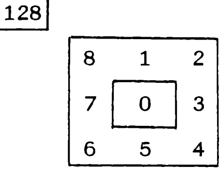
Die POT-Funktion ermöglicht es, die Stellung der Paddles festzustellen. Das Ergebnis liegt im Bereich von 0 bis 255. Die Ziffer hinter dem POT-Befehl gibt an, welches Paddle abgefragt werden soll.

11.5 JOY

Format: Variable = JOY
Ziel : Die Funktionen des JOYSTICK bestimmen.

Mit dem JOY-Befehl kann man feststellen, in welche Richtung der JOYSTICK zeigt, oder ob der Feuerknopf gedrückt wurde.

Feuer = 128



Beispiel:

```
100 REM ******************************
110 REM *** BEISPIEL:               ***
120 REM *** BEDIENUNG DES JOYSTICK  ***
130 REM ******************************
140 REM *** BILDSCHIRMSTARTADRESSE  ***
150 B= 1024
160 REM *** FARBADRESSE             ***
170 C=55296
180 REM *** POSITIONSAENDERUNG      ***
190 D=  420
200 PRINT"■"
210 REM *** JOYSTICK-ABFRAGE        ***
220 P=JOY
230 IF P=1 THEN D=D-40
240 IF P=2 THEN D=D-39
250 IF P=3 THEN D=D+ 1
260 IF P=4 THEN D=D+41
270 IF P=5 THEN D=D+40
280 IF P=6 THEN D=D+39
290 IF P=7 THEN D=D- 1
300 IF P=8 THEN D=D-41
310 REM *** BILDSCHIRM BEGRENZEN    ***
320 IF D<1 THEN D=1
330 IF D>999 THEN D=999
340 REM *** BILDSCHIRM LOESCHEN     ***
350 IF P=128 THEN GOTO 200
360 REM *** ZEICHEN POKEN           ***
370 POKE C+D,0
380 POKE B+D,102
390 GOTO 220
```

Art.-Nr. 56400

